Computer Graphics (UCS505)

Project on

Satellite Animation

Submitted By

Aashima	101917113
Harshita Gupta	101917125
Jaskaran Singh Purewal	101917129

B.E. Third Year – CSE5

Group No. -- 8

Submitted To:

Dr. Samya Muhuri

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

Patiala – 147001

Table of Contents

INTRODUCTION:

Satellite Animation is a simulation of take-off of the rocket which will deploy the satellite, deployment of the satellite and the subsequent orientation correction of its orbit around a planet. This graphics package is based on the OpenGL library functions. The programming language used here is C++ using OpenGL libraries.

The aim of this project is to provide an adept graphical visualization of deploying a satellite via a rocket, and its revolution around a planet about its set orbit.

You have 2 options in the menu at the title page:

By pressing S – key: Launch the satellite. (Start the animation)

By pressing Q – key: Quit the animation.

CONCEPTS:

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D Or 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly. Interactive graphics is the most important means of producing pictures since the invention of photography and television. We can make pictures of not only the real world objects but also of abstract objects such as mathematical surfaces on 4D and of data that have no inherent geometry.

Translation Function: A translation process moves every point a constant distance in a specified direction. It can be described as a rigid motion. A translation can also be interpreted as the addition of a constant vector to every point, or as shifting the origin of the coordinate system.

Scaling: It is used to alter or change the size of objects. The change is done using scaling factors. There are two scaling factors, i.e. Sx in x direction Sy in y-direction. If the original position is x and y. Scaling factors are Sx and Sy then the value of coordinates after scaling will be x1 and y1.

Animation: It is a method, where objects are manipulated to appear as moving, without the user input. It requires a form of infinite loop, which is usually achieved by a set of functions or class methods that describe the changes to each object in a small unit of time- called update or move.

Polygon : By specifying the vertices we have made different polygon , using GL_POLYGON from the GL/glut library. Lines: By using GL_LINES we have drawn lines in our project.

RGB Algorithm: The RGB Color Model is used for color representation, it is a color coordinate system having three primary colors, each primary color having its intensity value ranging from 0 to 1. Mixing these three primary colors at varying intensities produces a variety of colors.

USER-DEFINED FUNCTIONS:

1. void drawFilledCircle(GLfloat x, GLfloat y, GLfloat radius)

   This function is used to draw a filled circle.

2. void drawstring(int x, int y, char *s)

   This function will draw an input string s on the screen.

3. void semicircle(float radius,float u,float v)

   This function draws a semicircle of given radius and center on the screen.

4. void control()

   Determines the state of Satellite launch.

5. void stars()

   Twinkling stars animation.

6. void stars1()

   Different twinkling stars animation.

7. void static_rocket()

   Scene of the Rocket on ground.

8. void rocket_to_cam_pos()

   Rocket animation in the atmosphere.

9. void rocket_in_motion()

   Satellite animation in space.

10. void mars(float radius)

    Planet animation. (Mars)

11. void keyboard(unsigned char key, int x, int y)

    Manage keyboard inputs.

12. void page()

    Design of Title page.

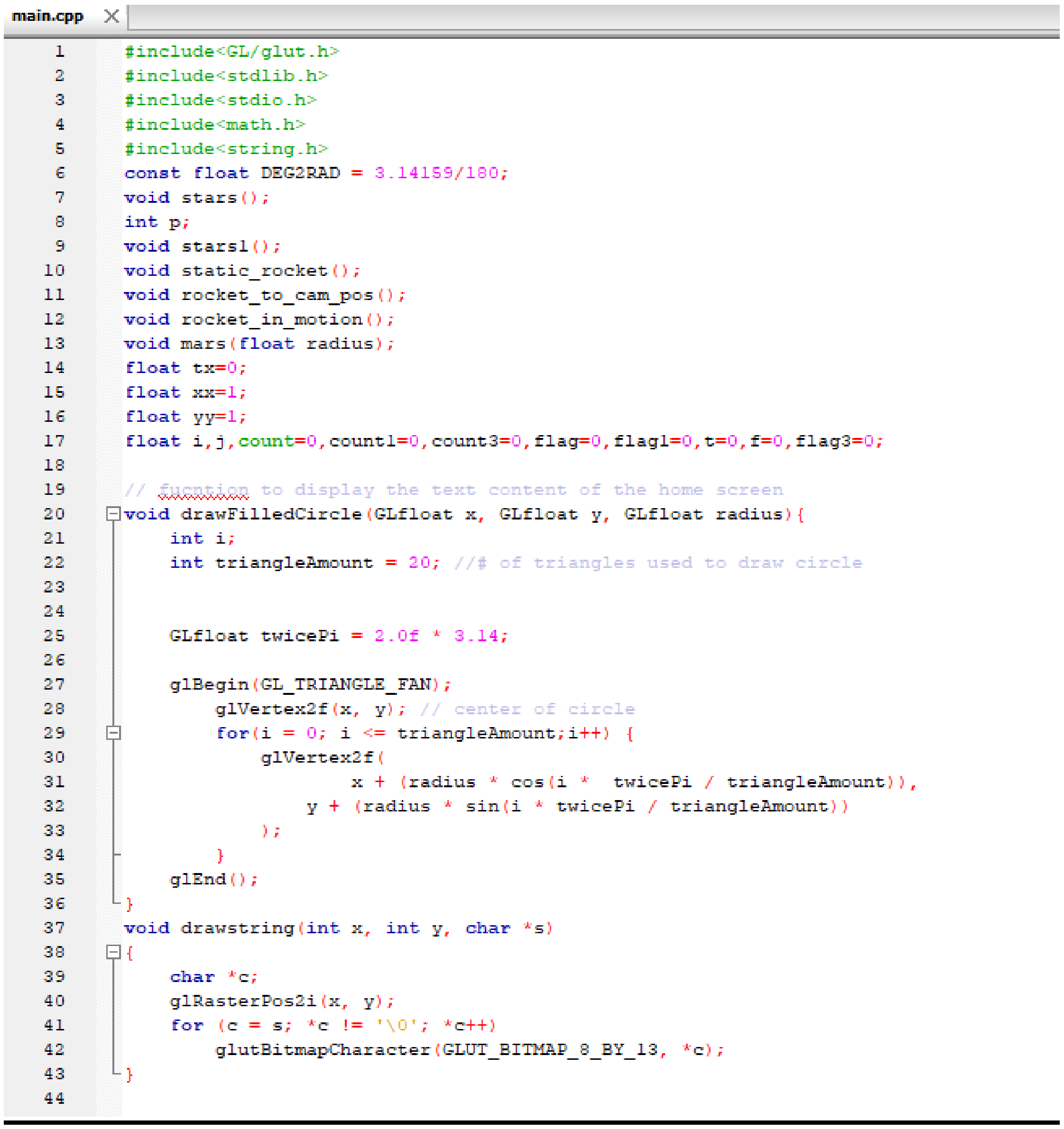
13. void display()

    Display all components.

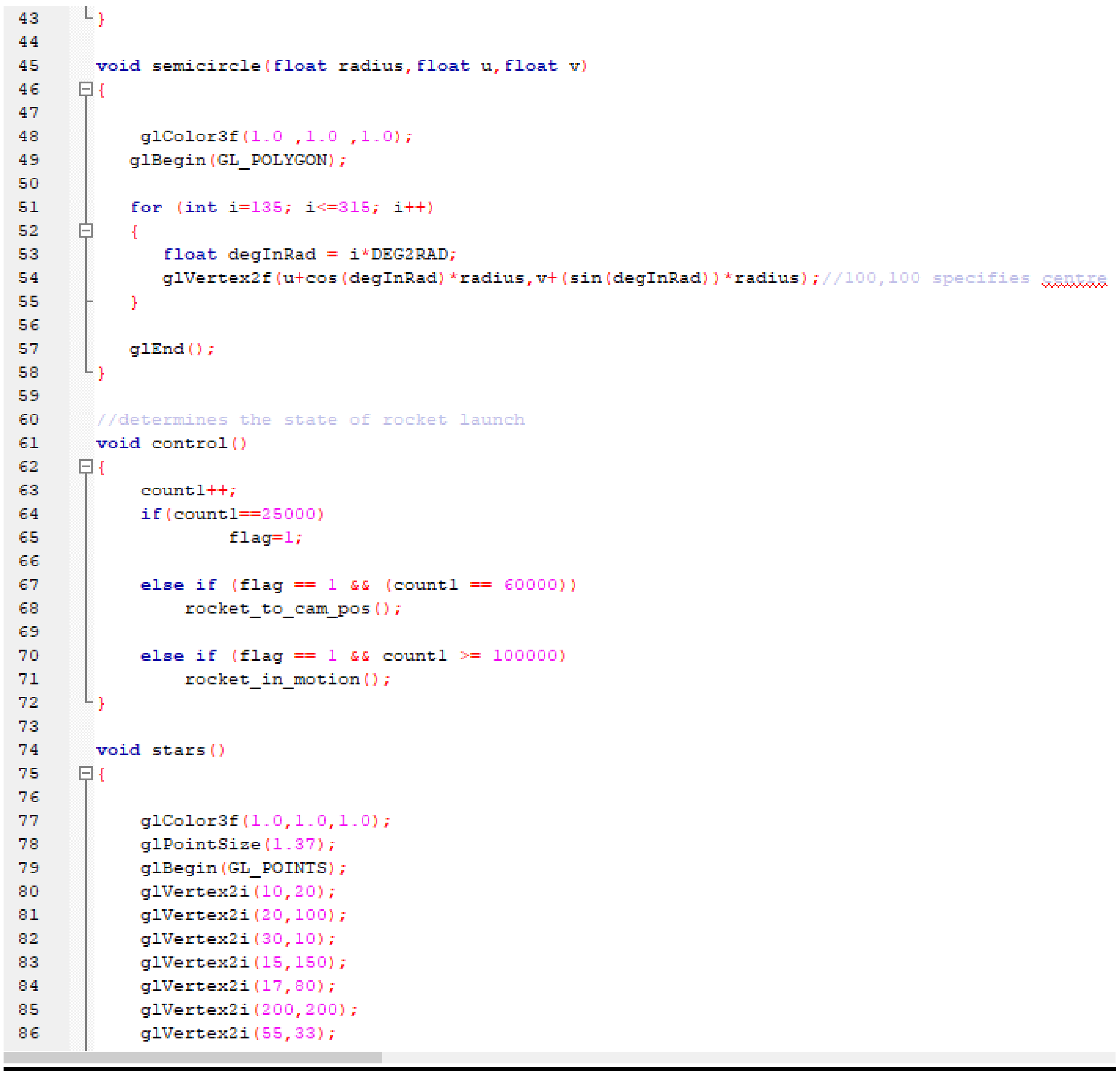
14. void myinit()

    Initialize window colors, mode, and point size.

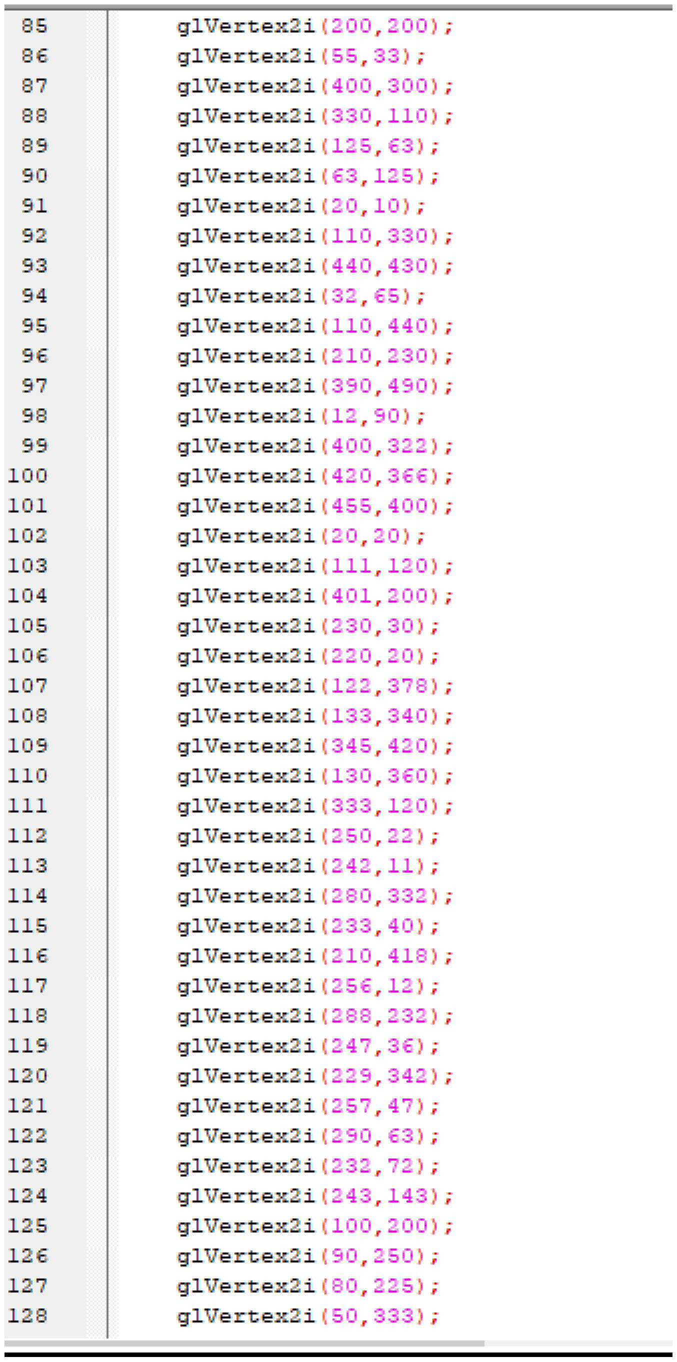
CODE:



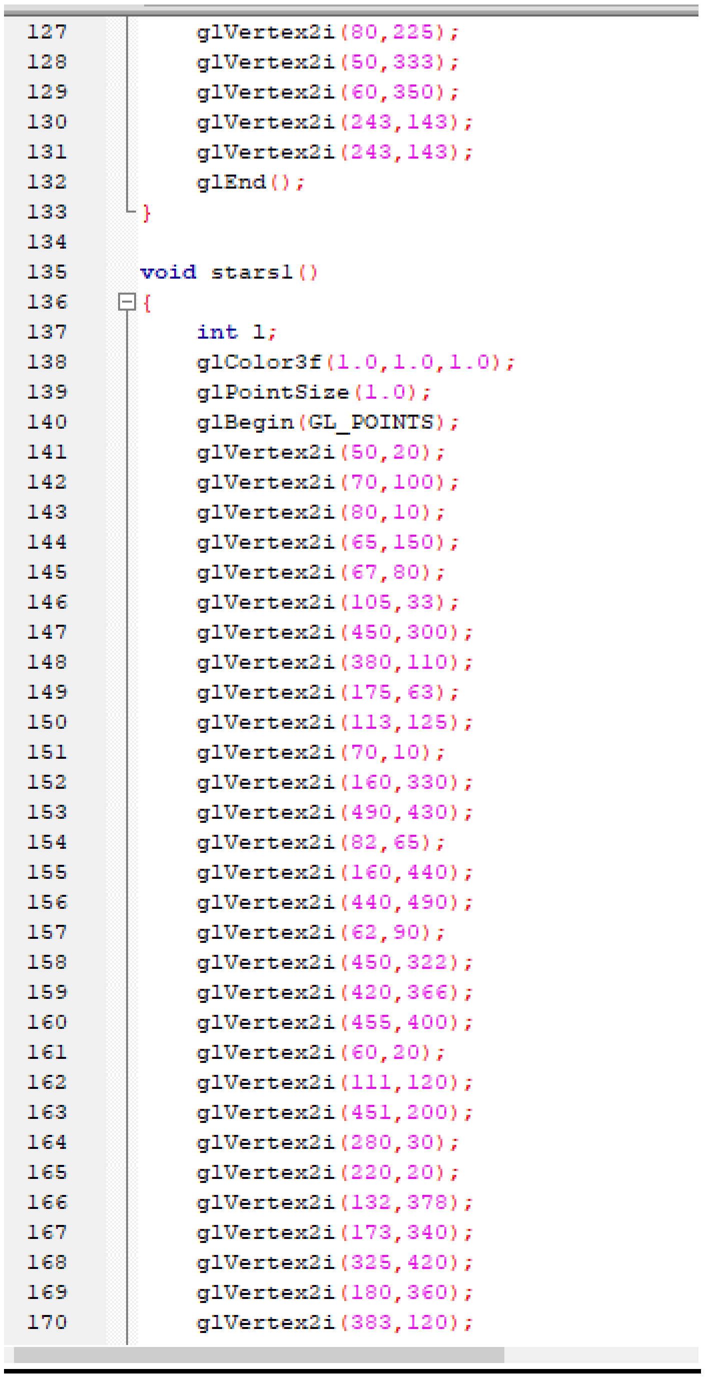
```
#include<GL/glut.h>
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
#include<string.h>
const float DEG2RAD = 3.14159/180;
void stars();
int p;
void stars1();
void static_rocket();
void rocket_to_cam_pos();
void rocket_in_motion();
void mars(float radius);
float tx=0;
float xx=1;
float yy=1;
float i,j,count=0,count1=0,count3=0,flag=0,flag1=0,t=0,f=0,flag3=0;

// fucntion to display the text content of the home screen
void drawFilledCircle(GLfloat x, GLfloat y, GLfloat radius){
    int i;
    int triangleAmount = 20; //# of triangles used to draw circle


    GLfloat twicePi = 2.0f * 3.14;

    glBegin(GL_TRIANGLE_FAN);
        glVertex2f(x, y); // center of circle
        for(i = 0; i <= triangleAmount;i++) {
            glVertex2f(
                    x + (radius * cos(i *  twicePi / triangleAmount)),
                y + (radius * sin(i * twicePi / triangleAmount))
            );
        }
    glEnd();
}
void drawstring(int x, int y, char *s)
{
    char *c;
    glRasterPos2i(x, y);
    for (c = s; *c != '\0'; *c++)
        glutBitmapCharacter(GLUT_BITMAP_8_BY_13, *c);
}

```



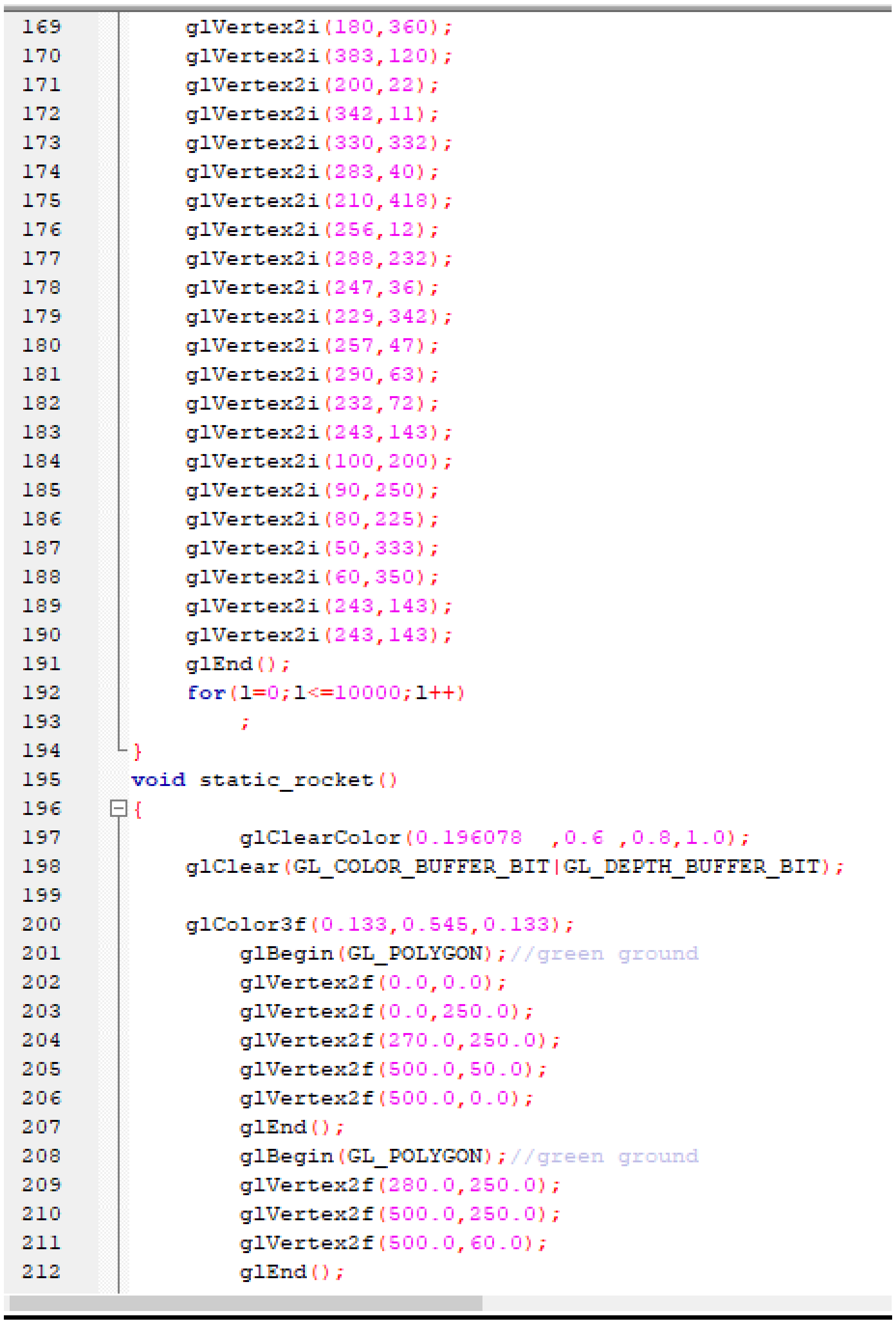
```
}

void semicircle(float radius,float u,float v)
{

     glColor3f(1.0 ,1.0 ,1.0);
    glBegin(GL_POLYGON);

    for (int i=135; i<=315; i++)
    {
       float degInRad = i*DEG2RAD;
       glVertex2f(u+cos(degInRad)*radius,v+(sin(degInRad))*radius);//100,100 specifies centre
    }

    glEnd();
}

//determines the state of rocket launch
void control()
{
     count1++;
     if(count1==25000)
             flag=1;

     else if (flag == 1 && (count1 == 60000))
         rocket_to_cam_pos();

     else if (flag == 1 && count1 >= 100000)
         rocket_in_motion();
}

void stars()
{

     glColor3f(1.0,1.0,1.0);
     glPointSize(1.37);
     glBegin(GL_POINTS);
     glVertex2i(10,20);
     glVertex2i(20,100);
     glVertex2i(30,10);
     glVertex2i(15,150);
     glVertex2i(17,80);
     glVertex2i(200,200);
     glVertex2i(55,33);
```



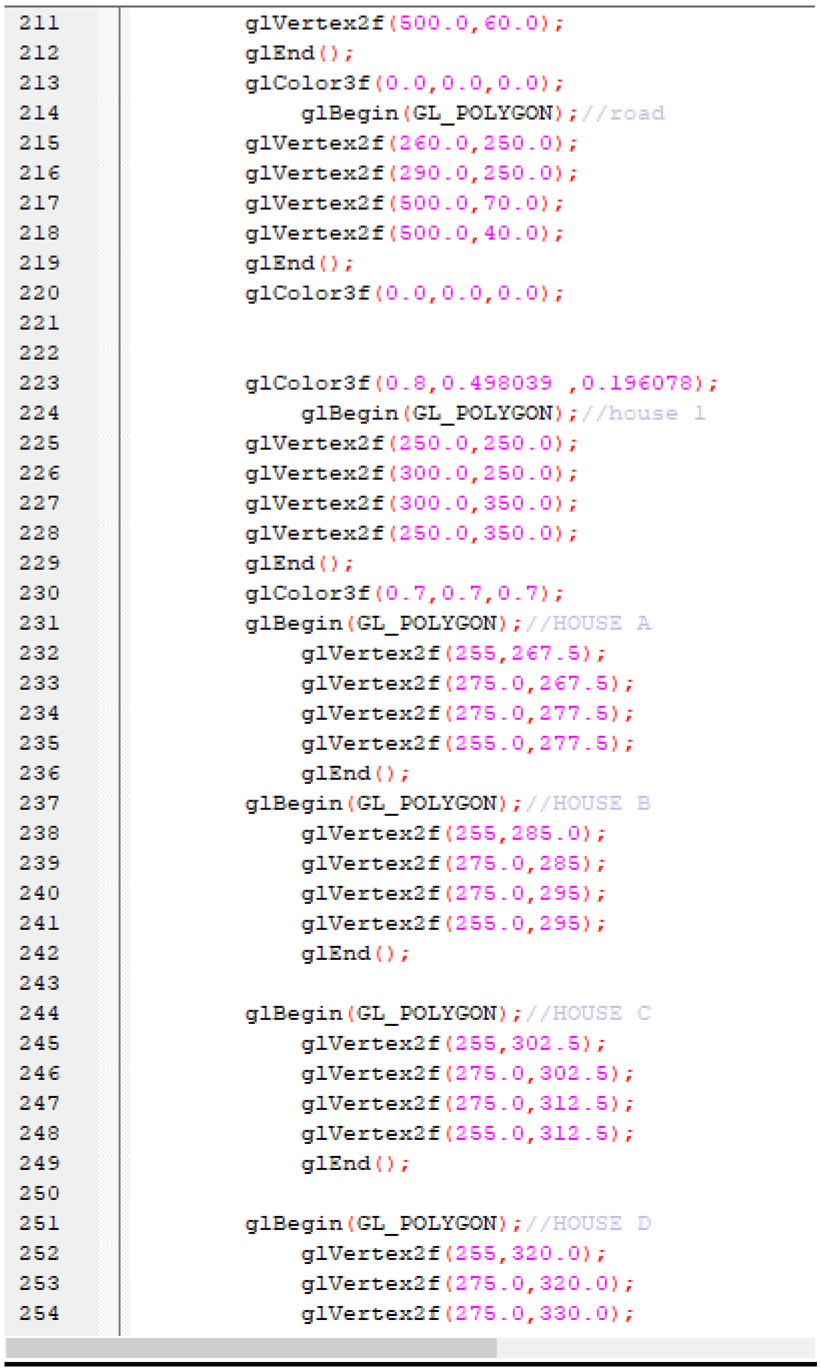
```
        glVertex2i(200,200);
        glVertex2i(55,33);
        glVertex2i(400,300);
        glVertex2i(330,110);
        glVertex2i(125,63);
        glVertex2i(63,125);
        glVertex2i(20,10);
        glVertex2i(110,330);
        glVertex2i(440,430);
        glVertex2i(32,65);
        glVertex2i(110,440);
        glVertex2i(210,230);
        glVertex2i(390,490);
        glVertex2i(12,90);
        glVertex2i(400,322);
        glVertex2i(420,366);
        glVertex2i(455,400);
        glVertex2i(20,20);
        glVertex2i(111,120);
        glVertex2i(401,200);
        glVertex2i(230,30);
        glVertex2i(220,20);
        glVertex2i(122,378);
        glVertex2i(133,340);
        glVertex2i(345,420);
        glVertex2i(130,360);
        glVertex2i(333,120);
        glVertex2i(250,22);
        glVertex2i(242,11);
        glVertex2i(280,332);
        glVertex2i(233,40);
        glVertex2i(210,418);
        glVertex2i(256,12);
        glVertex2i(288,232);
        glVertex2i(247,36);
        glVertex2i(229,342);
        glVertex2i(257,47);
        glVertex2i(290,63);
        glVertex2i(232,72);
        glVertex2i(243,143);
        glVertex2i(100,200);
        glVertex2i(90,250);
        glVertex2i(80,225);
        glVertex2i(50,333);
```



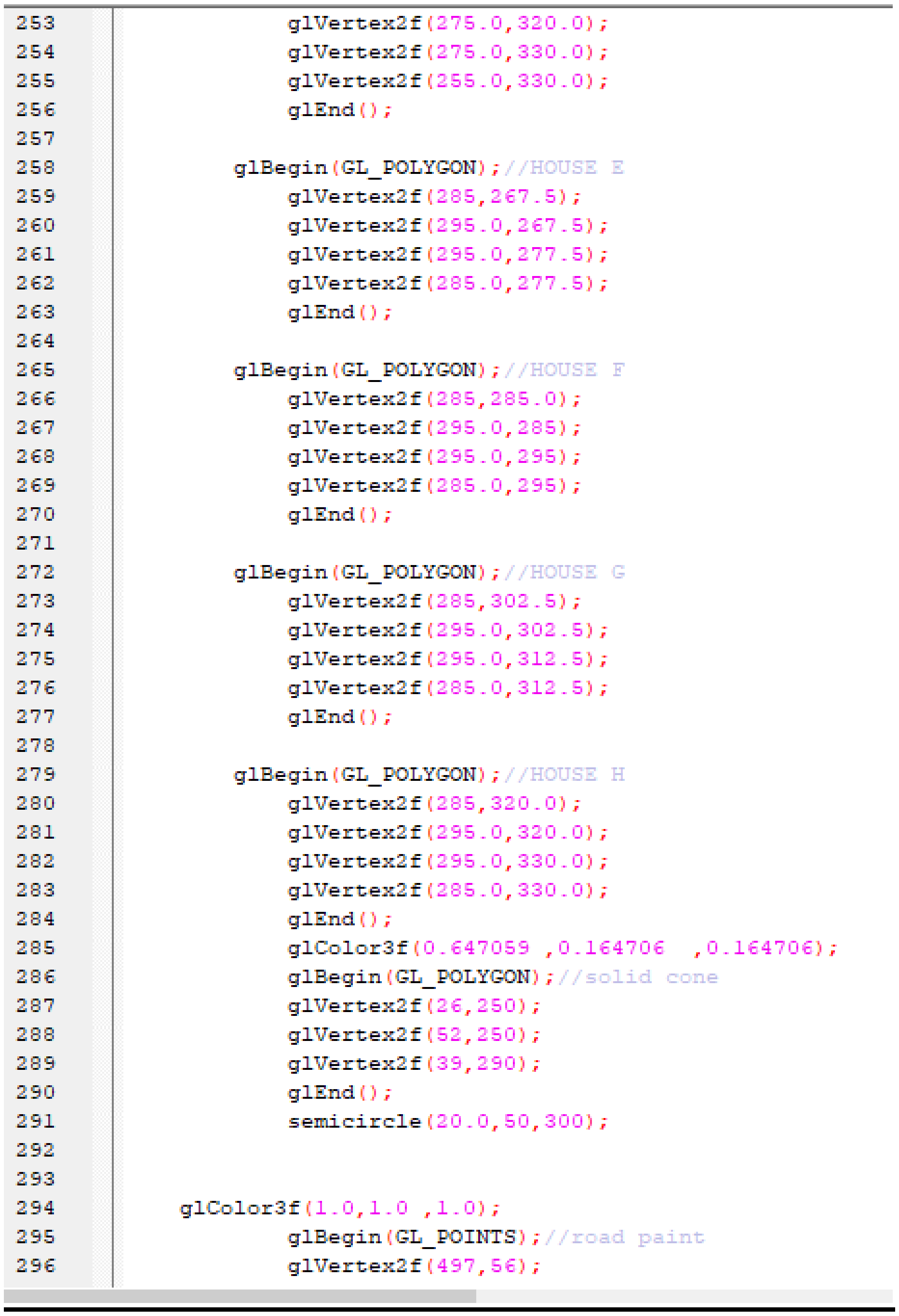
```
    glVertex2i(80,225);
    glVertex2i(50,333);
    glVertex2i(60,350);
    glVertex2i(243,143);
    glVertex2i(243,143);
    glEnd();
}

void stars1()
{
    int l;
    glColor3f(1.0,1.0,1.0);
    glPointSize(1.0);
    glBegin(GL_POINTS);
    glVertex2i(50,20);
    glVertex2i(70,100);
    glVertex2i(80,10);
    glVertex2i(65,150);
    glVertex2i(67,80);
    glVertex2i(105,33);
    glVertex2i(450,300);
    glVertex2i(380,110);
    glVertex2i(175,63);
    glVertex2i(113,125);
    glVertex2i(70,10);
    glVertex2i(160,330);
    glVertex2i(490,430);
    glVertex2i(82,65);
    glVertex2i(160,440);
    glVertex2i(440,490);
    glVertex2i(62,90);
    glVertex2i(450,322);
    glVertex2i(420,366);
    glVertex2i(455,400);
    glVertex2i(60,20);
    glVertex2i(111,120);
    glVertex2i(451,200);
    glVertex2i(280,30);
    glVertex2i(220,20);
    glVertex2i(132,378);
    glVertex2i(173,340);
    glVertex2i(325,420);
    glVertex2i(180,360);
    glVertex2i(383,120);
```



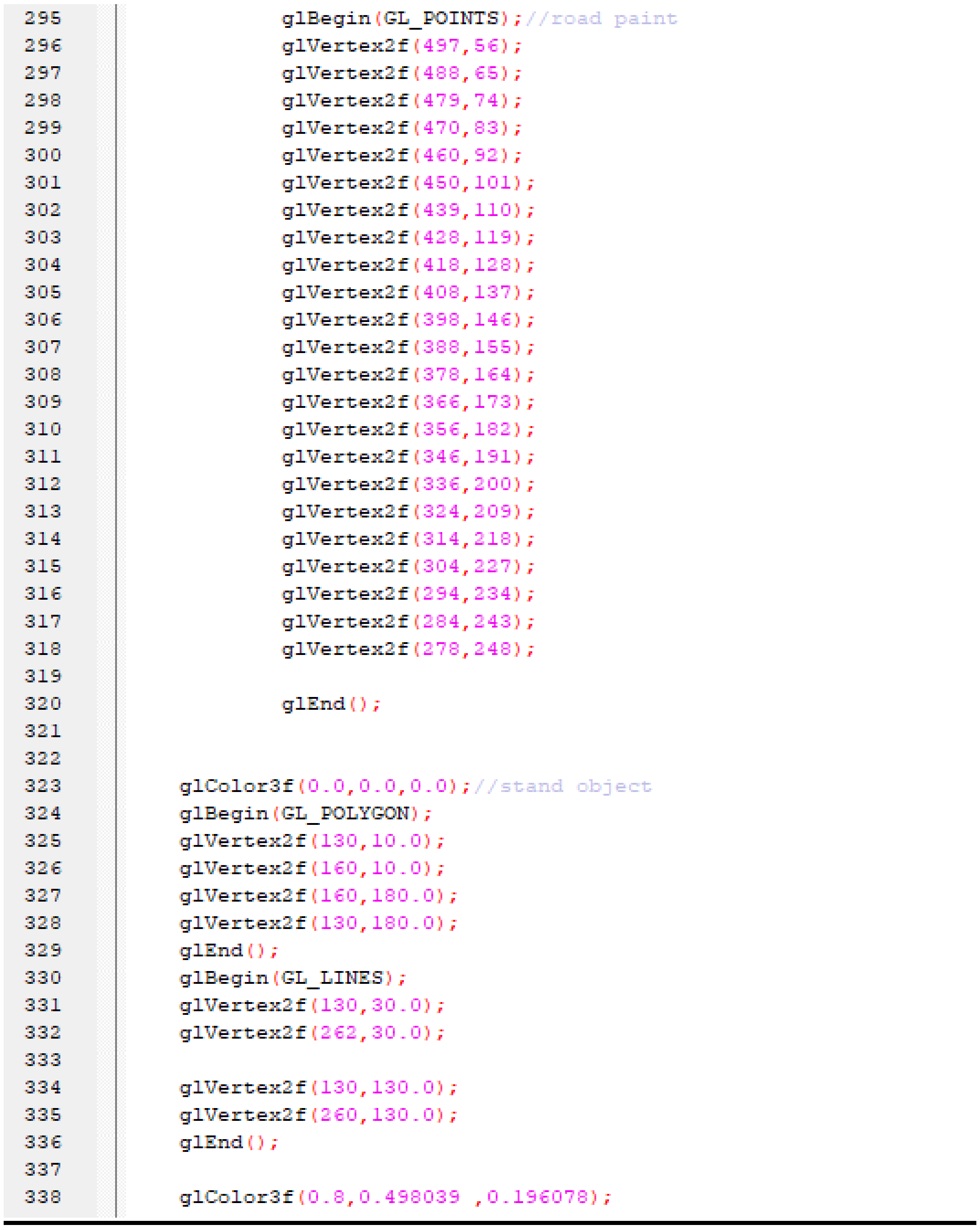
```
    glVertex2i(180,360);
    glVertex2i(383,120);
    glVertex2i(200,22);
    glVertex2i(342,11);
    glVertex2i(330,332);
    glVertex2i(283,40);
    glVertex2i(210,418);
    glVertex2i(256,12);
    glVertex2i(288,232);
    glVertex2i(247,36);
    glVertex2i(229,342);
    glVertex2i(257,47);
    glVertex2i(290,63);
    glVertex2i(232,72);
    glVertex2i(243,143);
    glVertex2i(100,200);
    glVertex2i(90,250);
    glVertex2i(80,225);
    glVertex2i(50,333);
    glVertex2i(60,350);
    glVertex2i(243,143);
    glVertex2i(243,143);
    glEnd();
    for(l=0;l<=10000;l++)
        ;
}
void static_rocket()
{
        glClearColor(0.196078  ,0.6 ,0.8,1.0);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

    glColor3f(0.133,0.545,0.133);
        glBegin(GL_POLYGON);//green ground
        glVertex2f(0.0,0.0);
        glVertex2f(0.0,250.0);
        glVertex2f(270.0,250.0);
        glVertex2f(500.0,50.0);
        glVertex2f(500.0,0.0);
        glEnd();
        glBegin(GL_POLYGON);//green ground
        glVertex2f(280.0,250.0);
        glVertex2f(500.0,250.0);
        glVertex2f(500.0,60.0);
        glEnd();
```



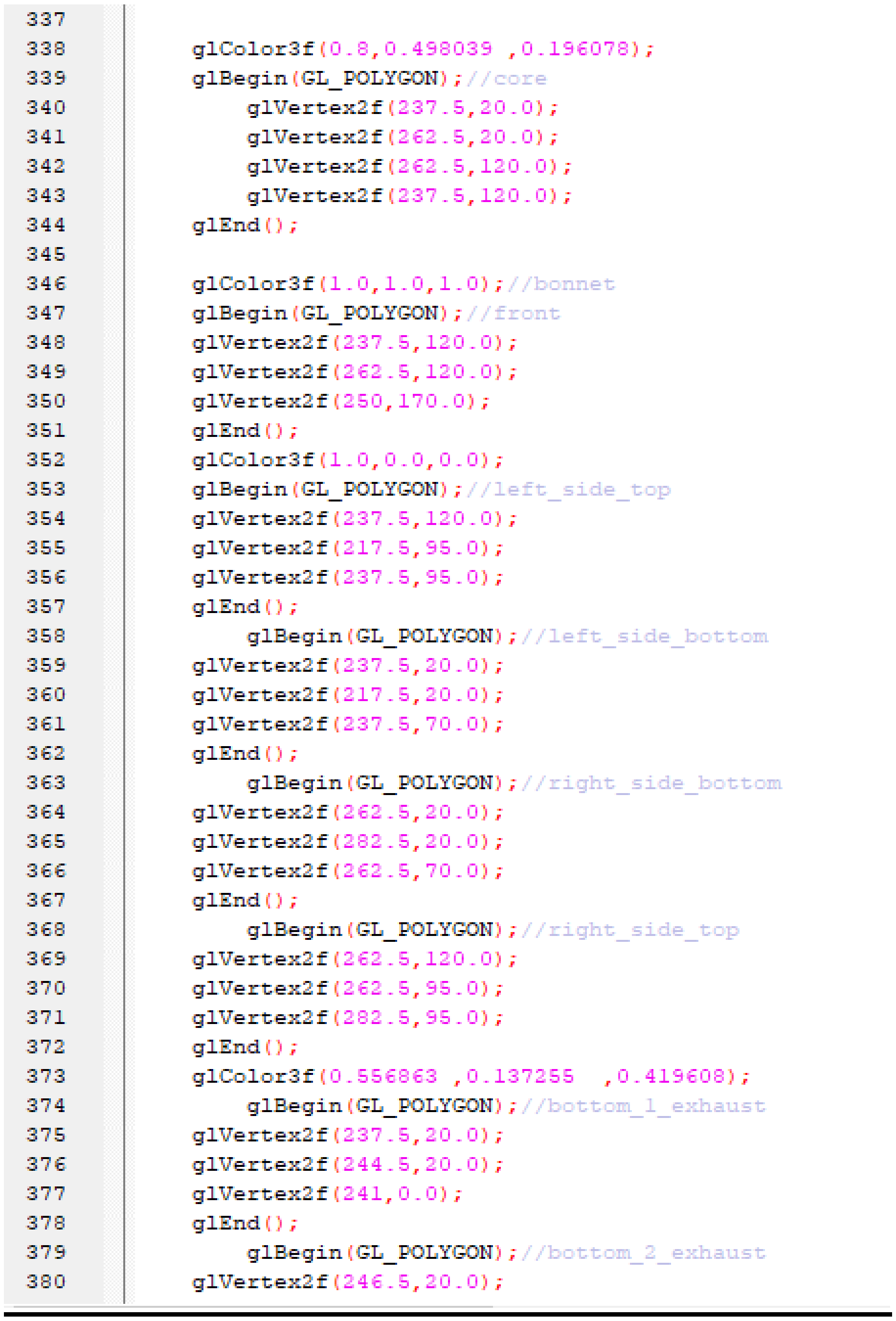
```
        glVertex2f(500.0,60.0);
        glEnd();
        glColor3f(0.0,0.0,0.0);
            glBegin(GL_POLYGON);//road
        glVertex2f(260.0,250.0);
        glVertex2f(290.0,250.0);
        glVertex2f(500.0,70.0);
        glVertex2f(500.0,40.0);
        glEnd();
        glColor3f(0.0,0.0,0.0);


        glColor3f(0.8,0.498039 ,0.196078);
            glBegin(GL_POLYGON);//house 1
        glVertex2f(250.0,250.0);
        glVertex2f(300.0,250.0);
        glVertex2f(300.0,350.0);
        glVertex2f(250.0,350.0);
        glEnd();
        glColor3f(0.7,0.7,0.7);
        glBegin(GL_POLYGON);//HOUSE A
            glVertex2f(255,267.5);
            glVertex2f(275.0,267.5);
            glVertex2f(275.0,277.5);
            glVertex2f(255.0,277.5);
            glEnd();
        glBegin(GL_POLYGON);//HOUSE B
            glVertex2f(255,285.0);
            glVertex2f(275.0,285);
            glVertex2f(275.0,295);
            glVertex2f(255.0,295);
            glEnd();

        glBegin(GL_POLYGON);//HOUSE C
            glVertex2f(255,302.5);
            glVertex2f(275.0,302.5);
            glVertex2f(275.0,312.5);
            glVertex2f(255.0,312.5);
            glEnd();

        glBegin(GL_POLYGON);//HOUSE D
            glVertex2f(255,320.0);
            glVertex2f(275.0,320.0);
            glVertex2f(275.0,330.0);
```



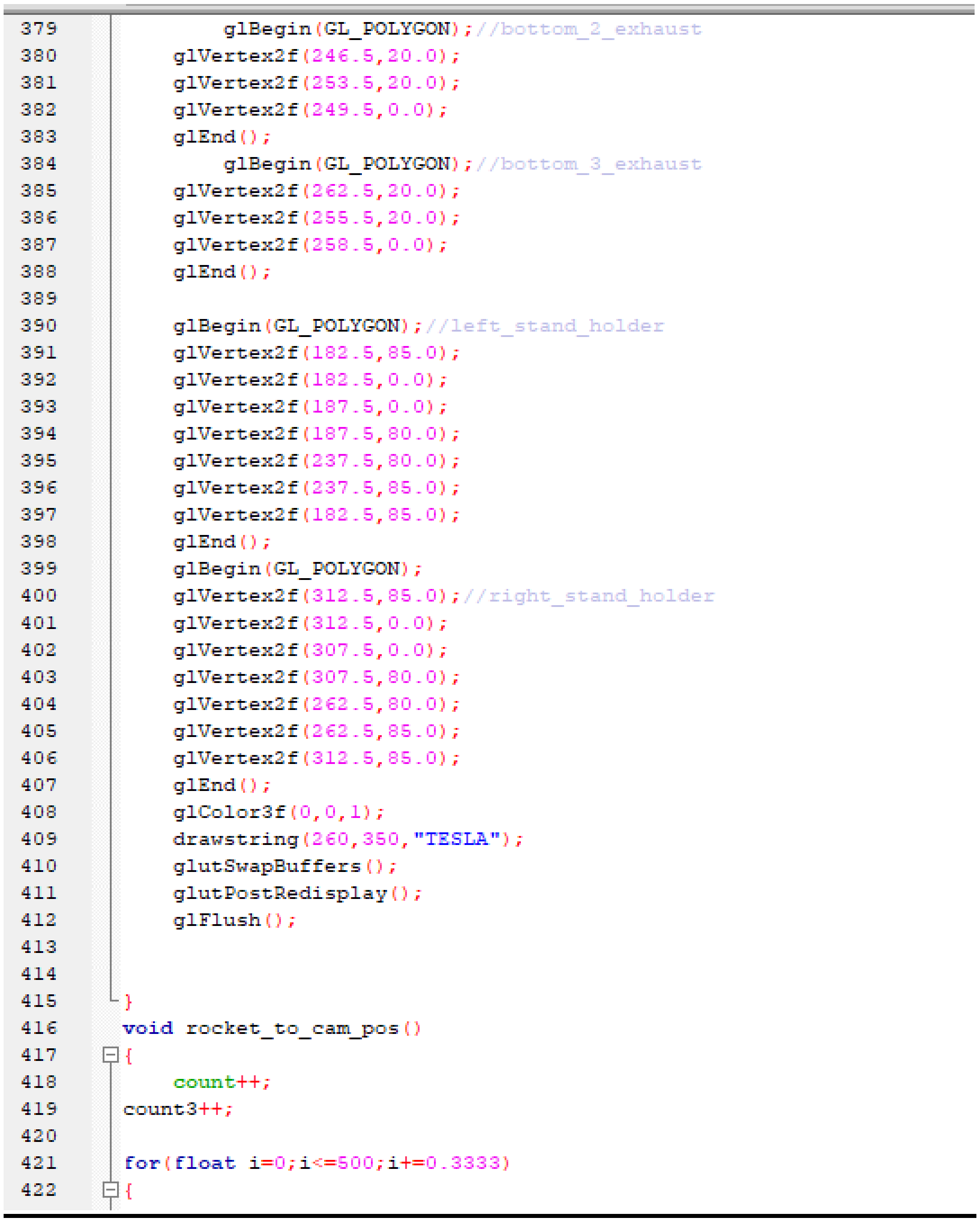
```
            glVertex2f(275.0,320.0);
            glVertex2f(275.0,330.0);
            glVertex2f(255.0,330.0);
            glEnd();

        glBegin(GL_POLYGON);//HOUSE E
            glVertex2f(285,267.5);
            glVertex2f(295.0,267.5);
            glVertex2f(295.0,277.5);
            glVertex2f(285.0,277.5);
            glEnd();

        glBegin(GL_POLYGON);//HOUSE F
            glVertex2f(285,285.0);
            glVertex2f(295.0,285);
            glVertex2f(295.0,295);
            glVertex2f(285.0,295);
            glEnd();

        glBegin(GL_POLYGON);//HOUSE G
            glVertex2f(285,302.5);
            glVertex2f(295.0,302.5);
            glVertex2f(295.0,312.5);
            glVertex2f(285.0,312.5);
            glEnd();

        glBegin(GL_POLYGON);//HOUSE H
            glVertex2f(285,320.0);
            glVertex2f(295.0,320.0);
            glVertex2f(295.0,330.0);
            glVertex2f(285.0,330.0);
            glEnd();
            glColor3f(0.647059 ,0.164706  ,0.164706);
            glBegin(GL_POLYGON);//solid cone
            glVertex2f(26,250);
            glVertex2f(52,250);
            glVertex2f(39,290);
            glEnd();
            semicircle(20.0,50,300);


    glColor3f(1.0,1.0 ,1.0);
            glBegin(GL_POINTS);//road paint
            glVertex2f(497,56);
```



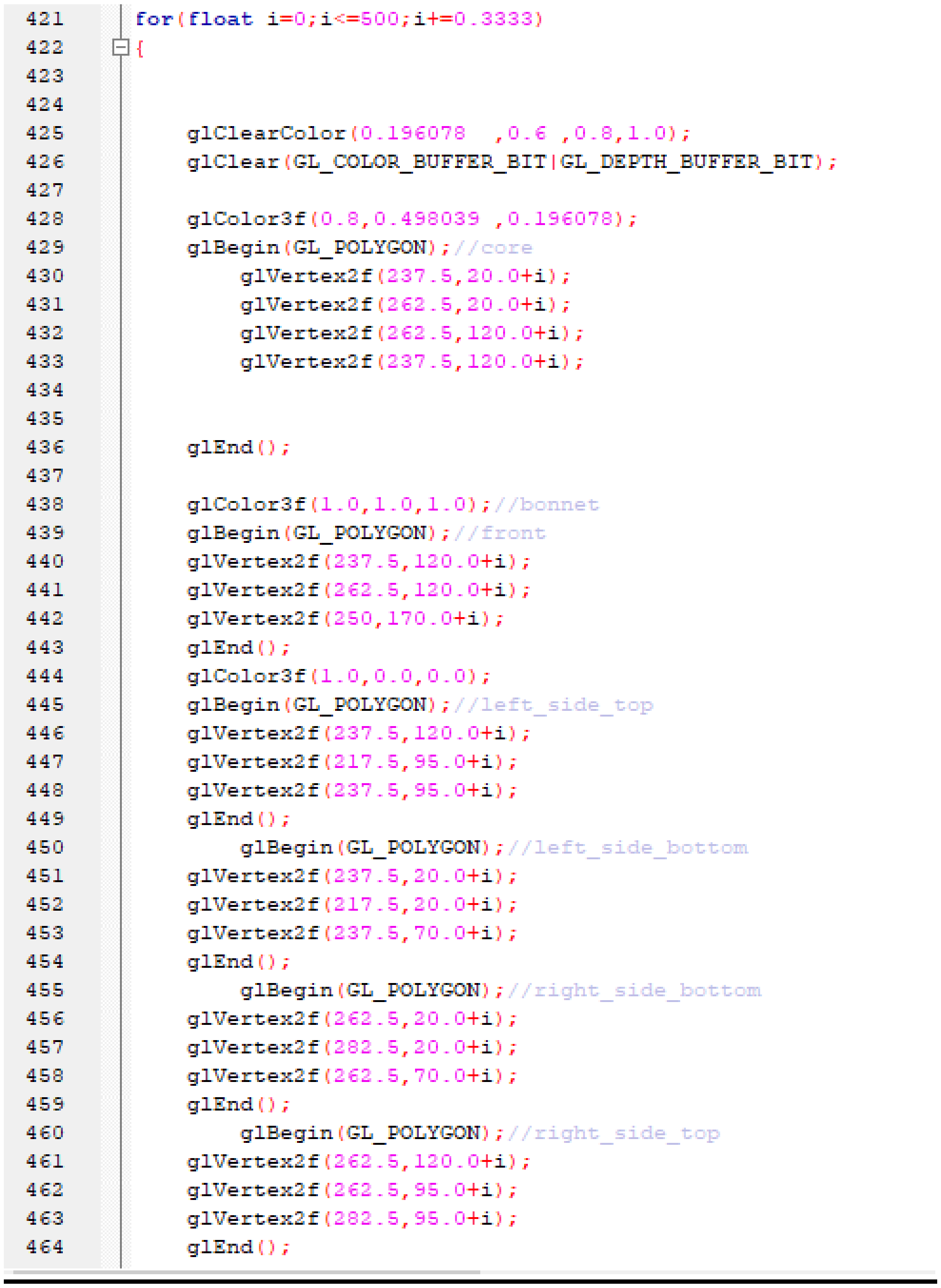
```
            glBegin(GL_POINTS);//road paint
            glVertex2f(497,56);
            glVertex2f(488,65);
            glVertex2f(479,74);
            glVertex2f(470,83);
            glVertex2f(460,92);
            glVertex2f(450,101);
            glVertex2f(439,110);
            glVertex2f(428,119);
            glVertex2f(418,128);
            glVertex2f(408,137);
            glVertex2f(398,146);
            glVertex2f(388,155);
            glVertex2f(378,164);
            glVertex2f(366,173);
            glVertex2f(356,182);
            glVertex2f(346,191);
            glVertex2f(336,200);
            glVertex2f(324,209);
            glVertex2f(314,218);
            glVertex2f(304,227);
            glVertex2f(294,234);
            glVertex2f(284,243);
            glVertex2f(278,248);

            glEnd();


    glColor3f(0.0,0.0,0.0);//stand object
    glBegin(GL_POLYGON);
    glVertex2f(130,10.0);
    glVertex2f(160,10.0);
    glVertex2f(160,180.0);
    glVertex2f(130,180.0);
    glEnd();
    glBegin(GL_LINES);
    glVertex2f(130,30.0);
    glVertex2f(262,30.0);

    glVertex2f(130,130.0);
    glVertex2f(260,130.0);
    glEnd();

    glColor3f(0.8,0.498039 ,0.196078);
```



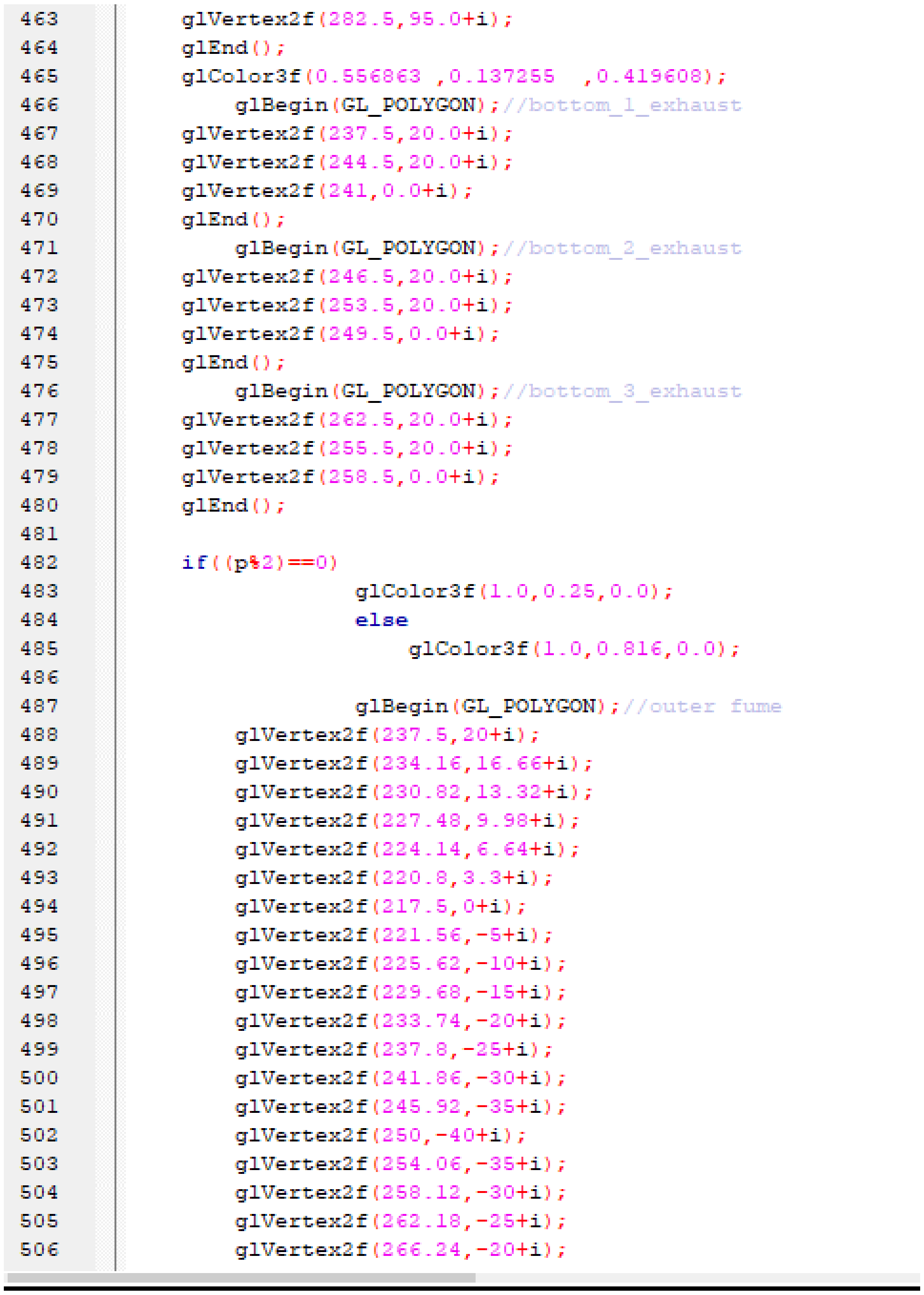
```

    glColor3f(0.8,0.498039 ,0.196078);
    glBegin(GL_POLYGON);//core
        glVertex2f(237.5,20.0);
        glVertex2f(262.5,20.0);
        glVertex2f(262.5,120.0);
        glVertex2f(237.5,120.0);
    glEnd();

    glColor3f(1.0,1.0,1.0);//bonnet
    glBegin(GL_POLYGON);//front
    glVertex2f(237.5,120.0);
    glVertex2f(262.5,120.0);
    glVertex2f(250,170.0);
    glEnd();
    glColor3f(1.0,0.0,0.0);
    glBegin(GL_POLYGON);//left_side_top
    glVertex2f(237.5,120.0);
    glVertex2f(217.5,95.0);
    glVertex2f(237.5,95.0);
    glEnd();
        glBegin(GL_POLYGON);//left_side_bottom
    glVertex2f(237.5,20.0);
    glVertex2f(217.5,20.0);
    glVertex2f(237.5,70.0);
    glEnd();
        glBegin(GL_POLYGON);//right_side_bottom
    glVertex2f(262.5,20.0);
    glVertex2f(282.5,20.0);
    glVertex2f(262.5,70.0);
    glEnd();
        glBegin(GL_POLYGON);//right_side_top
    glVertex2f(262.5,120.0);
    glVertex2f(262.5,95.0);
    glVertex2f(282.5,95.0);
    glEnd();
    glColor3f(0.556863 ,0.137255  ,0.419608);
        glBegin(GL_POLYGON);//bottom_1_exhaust
    glVertex2f(237.5,20.0);
    glVertex2f(244.5,20.0);
    glVertex2f(241,0.0);
    glEnd();
        glBegin(GL_POLYGON);//bottom_2_exhaust
    glVertex2f(246.5,20.0);
```



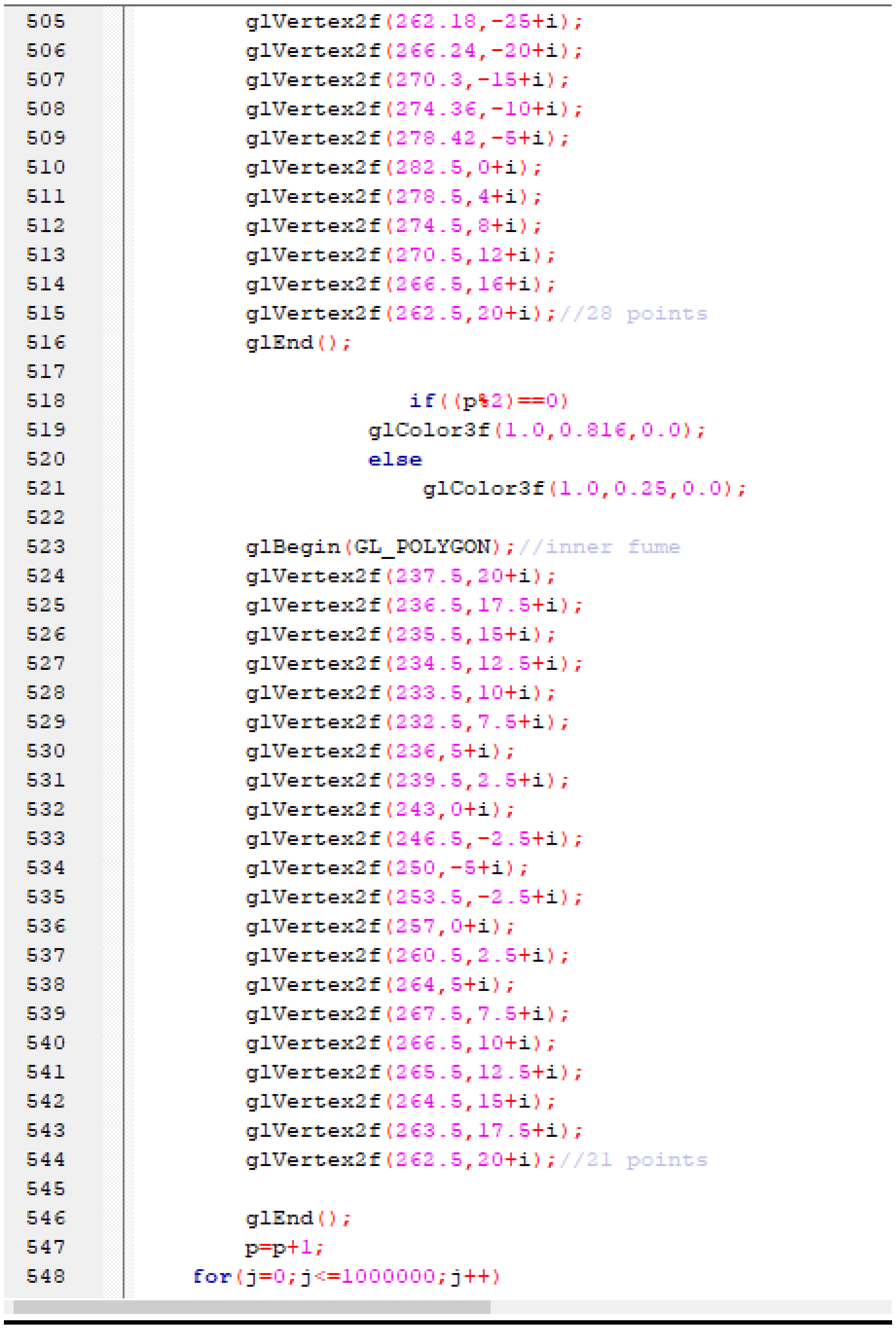
```
        glBegin(GL_POLYGON);//bottom_2_exhaust
    glVertex2f(246.5,20.0);
    glVertex2f(253.5,20.0);
    glVertex2f(249.5,0.0);
    glEnd();
        glBegin(GL_POLYGON);//bottom_3_exhaust
    glVertex2f(262.5,20.0);
    glVertex2f(255.5,20.0);
    glVertex2f(258.5,0.0);
    glEnd();

    glBegin(GL_POLYGON);//left_stand_holder
    glVertex2f(182.5,85.0);
    glVertex2f(182.5,0.0);
    glVertex2f(187.5,0.0);
    glVertex2f(187.5,80.0);
    glVertex2f(237.5,80.0);
    glVertex2f(237.5,85.0);
    glVertex2f(182.5,85.0);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex2f(312.5,85.0);//right_stand_holder
    glVertex2f(312.5,0.0);
    glVertex2f(307.5,0.0);
    glVertex2f(307.5,80.0);
    glVertex2f(262.5,80.0);
    glVertex2f(262.5,85.0);
    glVertex2f(312.5,85.0);
    glEnd();
    glColor3f(0,0,1);
    drawstring(260,350,"TESLA");
    glutSwapBuffers();
    glutPostRedisplay();
    glFlush();


}
void rocket_to_cam_pos()
{
    count++;
count3++;

for(float i=0;i<=500;i+=0.3333)
{
```



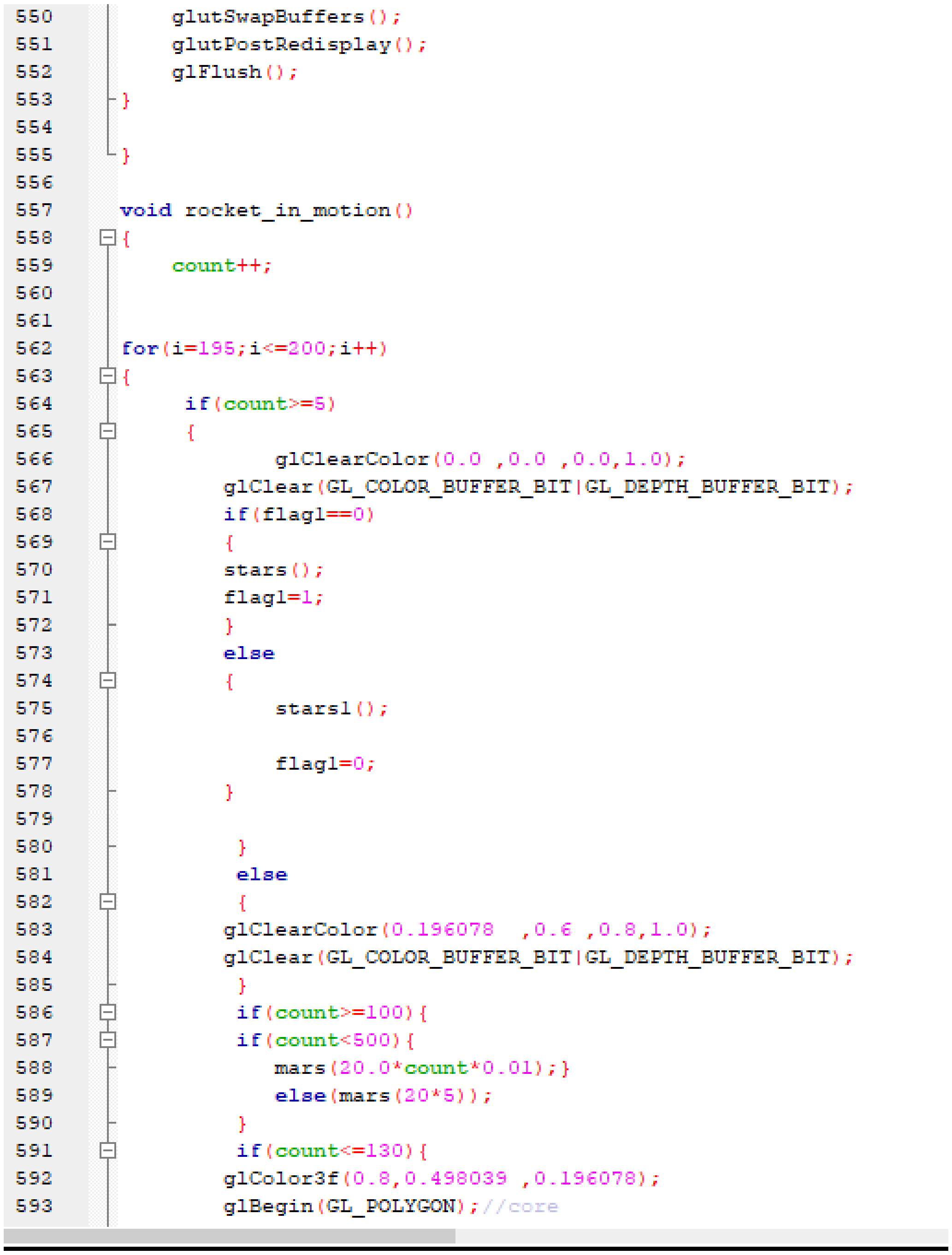
```
for(float i=0;i<=500;i+=0.3333)
{


    glClearColor(0.196078  ,0.6 ,0.8,1.0);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

    glColor3f(0.8,0.498039 ,0.196078);
    glBegin(GL_POLYGON);//core
        glVertex2f(237.5,20.0+i);
        glVertex2f(262.5,20.0+i);
        glVertex2f(262.5,120.0+i);
        glVertex2f(237.5,120.0+i);


    glEnd();

    glColor3f(1.0,1.0,1.0);//bonnet
    glBegin(GL_POLYGON);//front
    glVertex2f(237.5,120.0+i);
    glVertex2f(262.5,120.0+i);
    glVertex2f(250,170.0+i);
    glEnd();
    glColor3f(1.0,0.0,0.0);
    glBegin(GL_POLYGON);//left_side_top
    glVertex2f(237.5,120.0+i);
    glVertex2f(217.5,95.0+i);
    glVertex2f(237.5,95.0+i);
    glEnd();
        glBegin(GL_POLYGON);//left_side_bottom
    glVertex2f(237.5,20.0+i);
    glVertex2f(217.5,20.0+i);
    glVertex2f(237.5,70.0+i);
    glEnd();
        glBegin(GL_POLYGON);//right_side_bottom
    glVertex2f(262.5,20.0+i);
    glVertex2f(282.5,20.0+i);
    glVertex2f(262.5,70.0+i);
    glEnd();
        glBegin(GL_POLYGON);//right_side_top
    glVertex2f(262.5,120.0+i);
    glVertex2f(262.5,95.0+i);
    glVertex2f(282.5,95.0+i);
    glEnd();
```

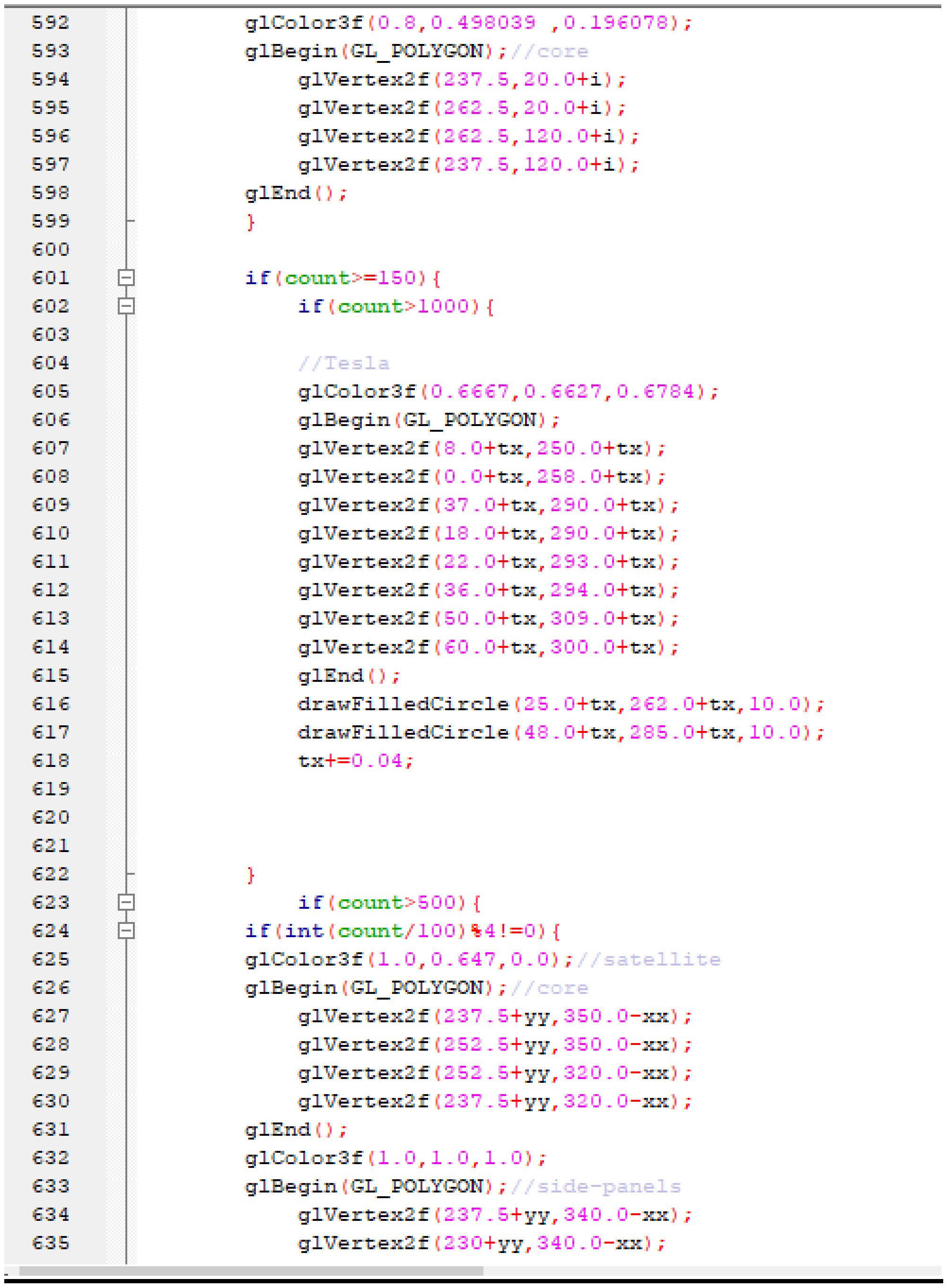


```
glVertex2f(282.5,95.0+i);
glEnd();
glColor3f(0.556863 ,0.137255  ,0.419608);
    glBegin(GL_POLYGON);//bottom_1_exhaust
glVertex2f(237.5,20.0+i);
glVertex2f(244.5,20.0+i);
glVertex2f(241,0.0+i);
glEnd();
    glBegin(GL_POLYGON);//bottom_2_exhaust
glVertex2f(246.5,20.0+i);
glVertex2f(253.5,20.0+i);
glVertex2f(249.5,0.0+i);
glEnd();
    glBegin(GL_POLYGON);//bottom_3_exhaust
glVertex2f(262.5,20.0+i);
glVertex2f(255.5,20.0+i);
glVertex2f(258.5,0.0+i);
glEnd();

if((p%2)==0)
             glColor3f(1.0,0.25,0.0);
             else
                 glColor3f(1.0,0.816,0.0);

             glBegin(GL_POLYGON);//outer fume
    glVertex2f(237.5,20+i);
    glVertex2f(234.16,16.66+i);
    glVertex2f(230.82,13.32+i);
    glVertex2f(227.48,9.98+i);
    glVertex2f(224.14,6.64+i);
    glVertex2f(220.8,3.3+i);
    glVertex2f(217.5,0+i);
    glVertex2f(221.56,-5+i);
    glVertex2f(225.62,-10+i);
    glVertex2f(229.68,-15+i);
    glVertex2f(233.74,-20+i);
    glVertex2f(237.8,-25+i);
    glVertex2f(241.86,-30+i);
    glVertex2f(245.92,-35+i);
    glVertex2f(250,-40+i);
    glVertex2f(254.06,-35+i);
    glVertex2f(258.12,-30+i);
    glVertex2f(262.18,-25+i);
    glVertex2f(266.24,-20+i);
```

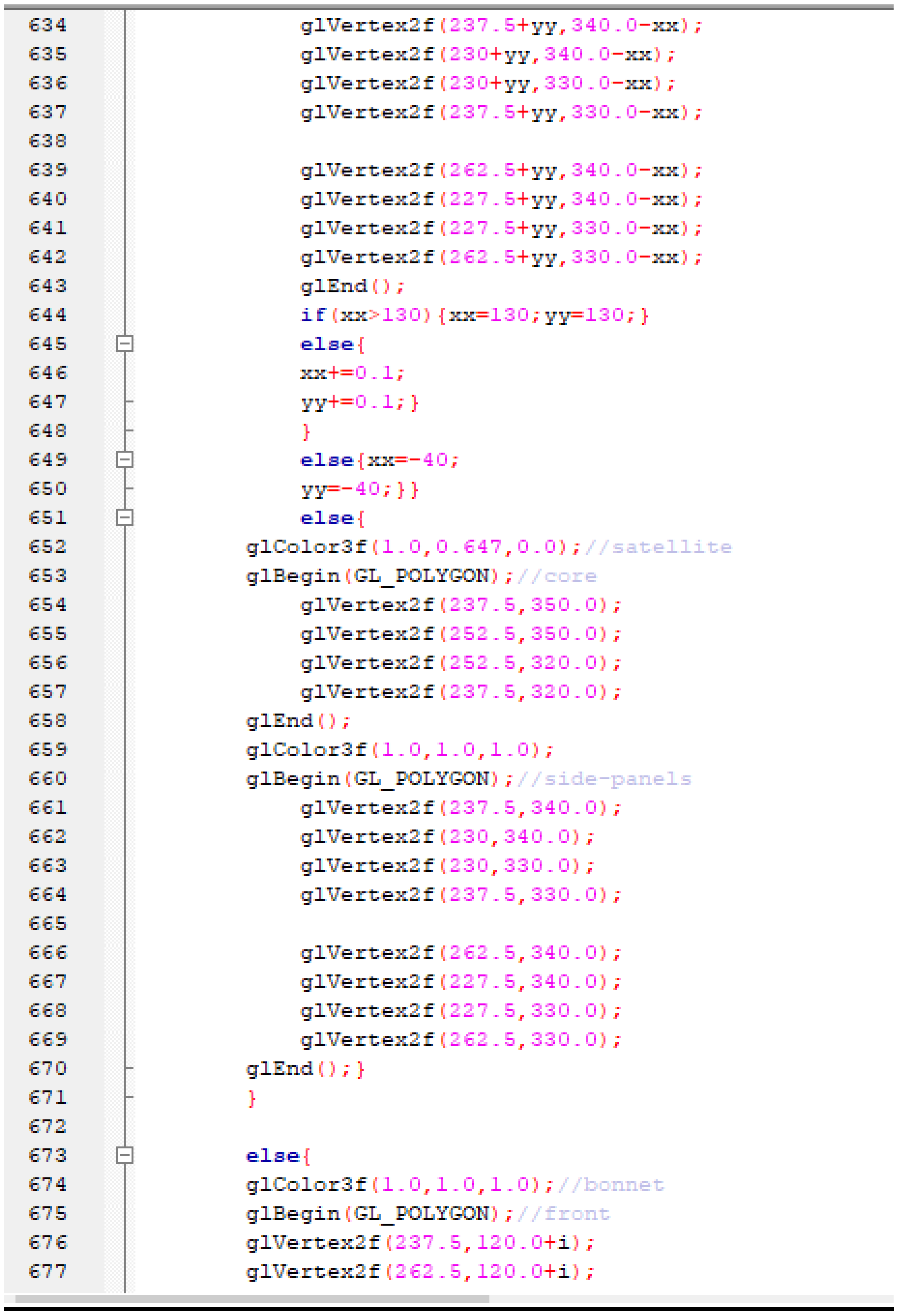


```
        glVertex2f(262.18,-25+i);
        glVertex2f(266.24,-20+i);
        glVertex2f(270.3,-15+i);
        glVertex2f(274.36,-10+i);
        glVertex2f(278.42,-5+i);
        glVertex2f(282.5,0+i);
        glVertex2f(278.5,4+i);
        glVertex2f(274.5,8+i);
        glVertex2f(270.5,12+i);
        glVertex2f(266.5,16+i);
        glVertex2f(262.5,20+i);//28 points
        glEnd();

                    if((p%2)==0)
                 glColor3f(1.0,0.816,0.0);
                 else
                     glColor3f(1.0,0.25,0.0);

        glBegin(GL_POLYGON);//inner fume
        glVertex2f(237.5,20+i);
        glVertex2f(236.5,17.5+i);
        glVertex2f(235.5,15+i);
        glVertex2f(234.5,12.5+i);
        glVertex2f(233.5,10+i);
        glVertex2f(232.5,7.5+i);
        glVertex2f(236,5+i);
        glVertex2f(239.5,2.5+i);
        glVertex2f(243,0+i);
        glVertex2f(246.5,-2.5+i);
        glVertex2f(250,-5+i);
        glVertex2f(253.5,-2.5+i);
        glVertex2f(257,0+i);
        glVertex2f(260.5,2.5+i);
        glVertex2f(264,5+i);
        glVertex2f(267.5,7.5+i);
        glVertex2f(266.5,10+i);
        glVertex2f(265.5,12.5+i);
        glVertex2f(264.5,15+i);
        glVertex2f(263.5,17.5+i);
        glVertex2f(262.5,20+i);//21 points

        glEnd();
        p=p+1;
    for(j=0;j<=1000000;j++)
```

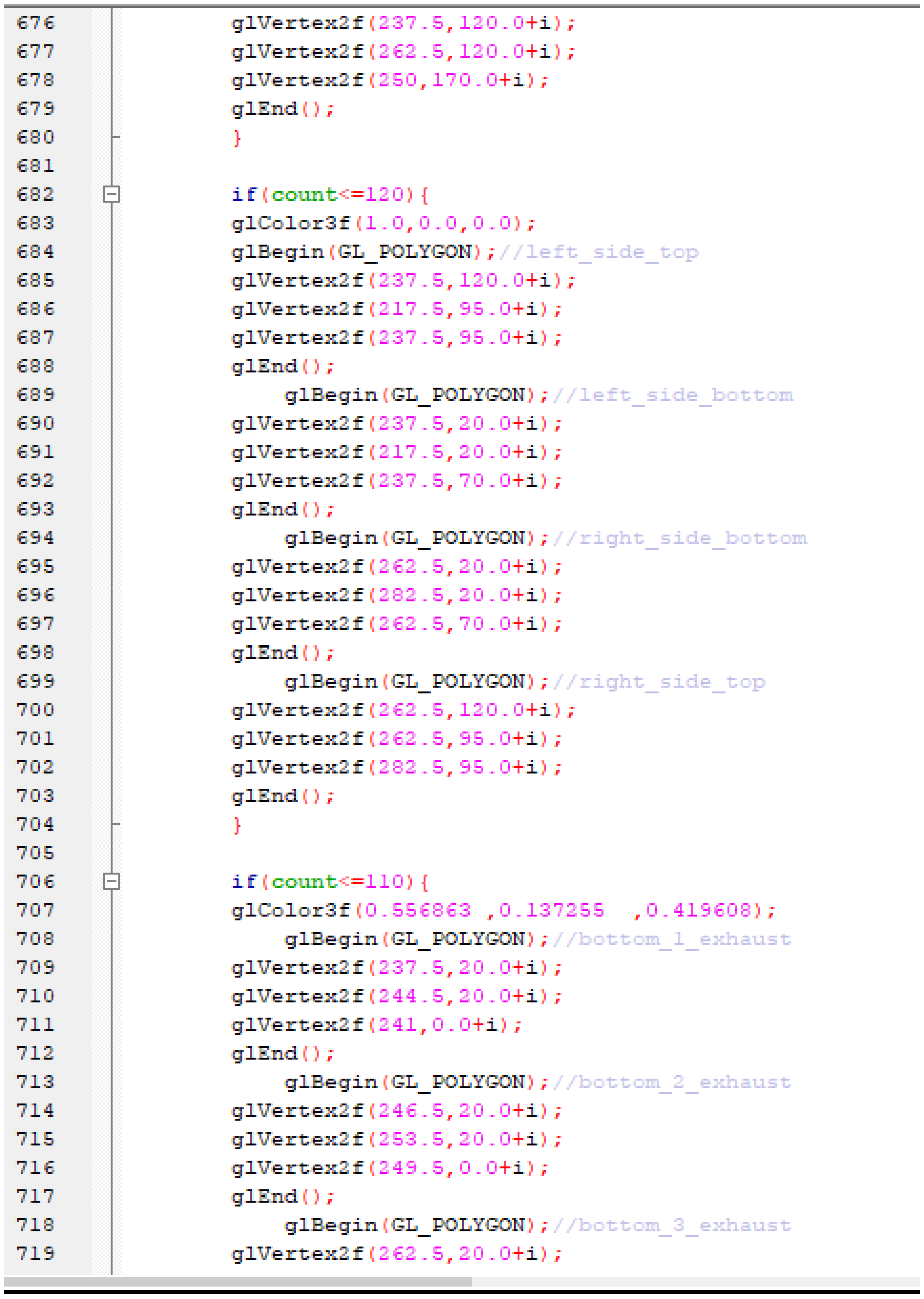

```
    glutSwapBuffers();
    glutPostRedisplay();
    glFlush();
}

}

void rocket_in_motion()
{
    count++;


for(i=195;i<=200;i++)
{
     if(count>=5)
     {
            glClearColor(0.0 ,0.0 ,0.0,1.0);
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        if(flag1==0)
        {
        stars();
        flag1=1;
        }
        else
        {
            stars1();

            flag1=0;
        }

         }
         else
         {
        glClearColor(0.196078  ,0.6 ,0.8,1.0);
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
         }
         if(count>=100){
         if(count<500){
            mars(20.0*count*0.01);}
            else(mars(20*5));
         }
         if(count<=130){
        glColor3f(0.8,0.498039 ,0.196078);
        glBegin(GL_POLYGON);//core
```



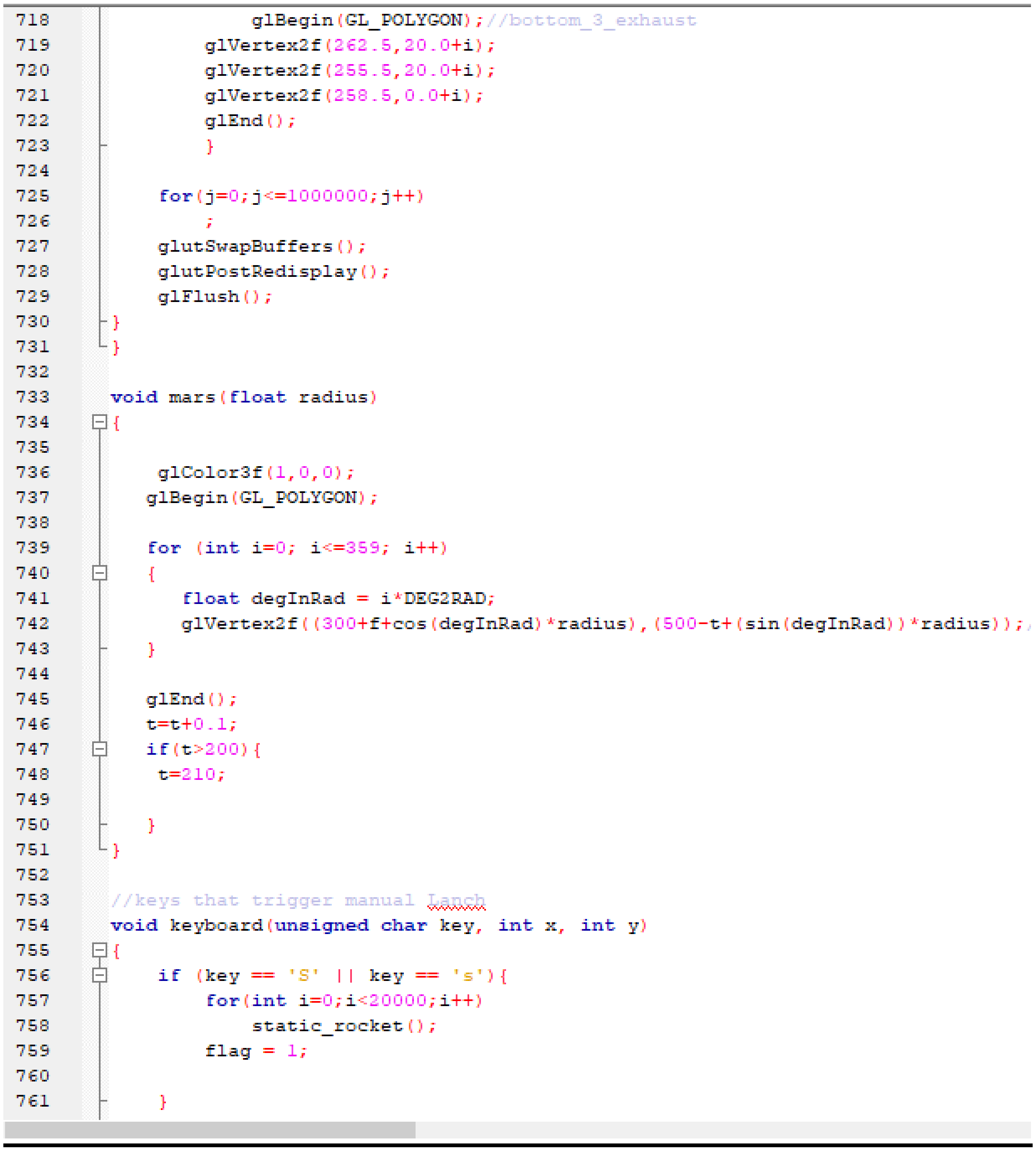
```
        glColor3f(0.8,0.498039 ,0.196078);
        glBegin(GL_POLYGON);//core
            glVertex2f(237.5,20.0+i);
            glVertex2f(262.5,20.0+i);
            glVertex2f(262.5,120.0+i);
            glVertex2f(237.5,120.0+i);
        glEnd();
        }

        if(count>=150){
            if(count>1000){

            //Tesla
            glColor3f(0.6667,0.6627,0.6784);
            glBegin(GL_POLYGON);
            glVertex2f(8.0+tx,250.0+tx);
            glVertex2f(0.0+tx,258.0+tx);
            glVertex2f(37.0+tx,290.0+tx);
            glVertex2f(18.0+tx,290.0+tx);
            glVertex2f(22.0+tx,293.0+tx);
            glVertex2f(36.0+tx,294.0+tx);
            glVertex2f(50.0+tx,309.0+tx);
            glVertex2f(60.0+tx,300.0+tx);
            glEnd();
            drawFilledCircle(25.0+tx,262.0+tx,10.0);
            drawFilledCircle(48.0+tx,285.0+tx,10.0);
            tx+=0.04;



        }
            if(count>500){
        if(int(count/100)%4!=0){
        glColor3f(1.0,0.647,0.0);//satellite
        glBegin(GL_POLYGON);//core
            glVertex2f(237.5+yy,350.0-xx);
            glVertex2f(252.5+yy,350.0-xx);
            glVertex2f(252.5+yy,320.0-xx);
            glVertex2f(237.5+yy,320.0-xx);
        glEnd();
        glColor3f(1.0,1.0,1.0);
        glBegin(GL_POLYGON);//side-panels
            glVertex2f(237.5+yy,340.0-xx);
            glVertex2f(230+yy,340.0-xx);
```



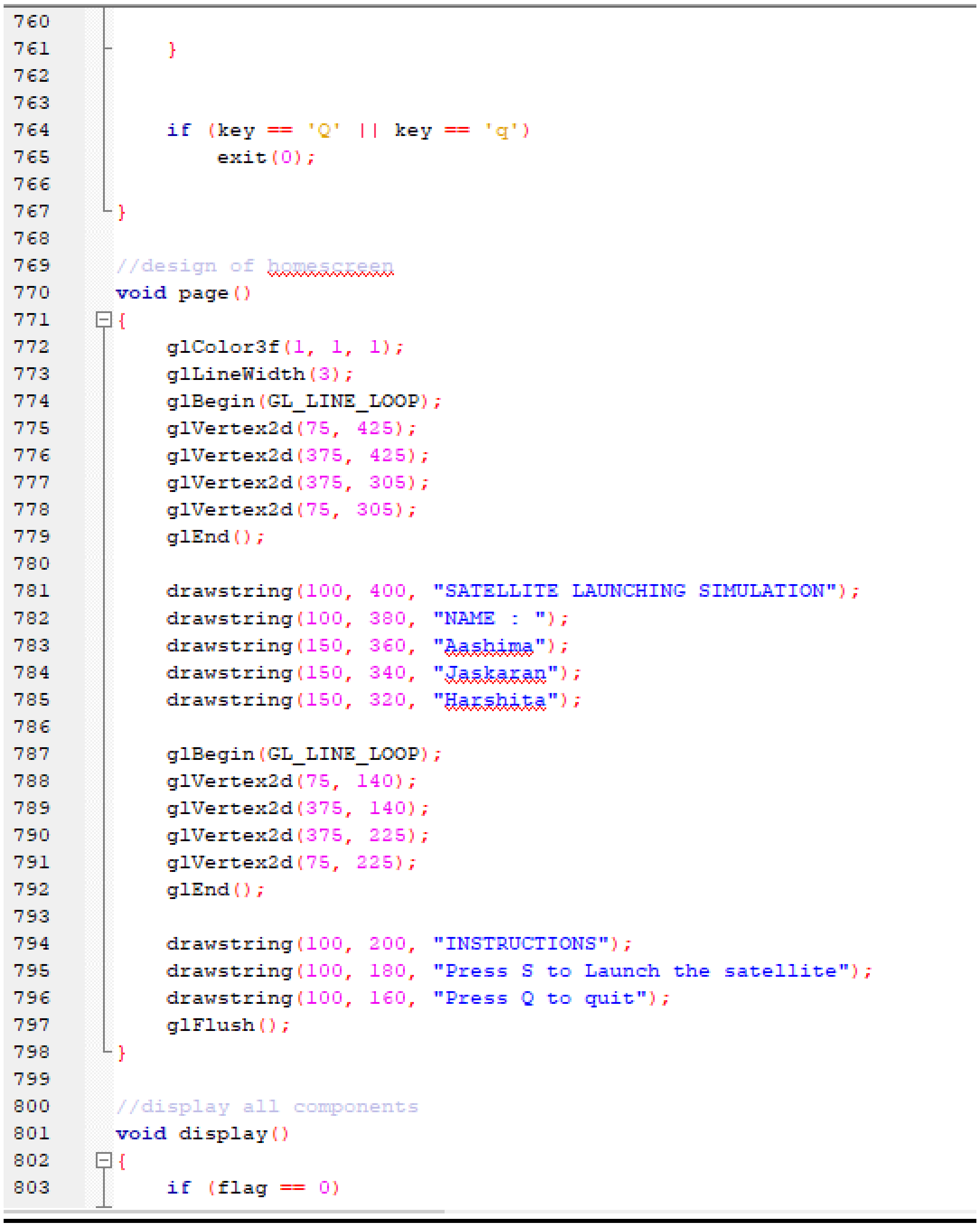
```
                glVertex2f(237.5+yy,340.0-xx);
                glVertex2f(230+yy,340.0-xx);
                glVertex2f(230+yy,330.0-xx);
                glVertex2f(237.5+yy,330.0-xx);

                glVertex2f(262.5+yy,340.0-xx);
                glVertex2f(227.5+yy,340.0-xx);
                glVertex2f(227.5+yy,330.0-xx);
                glVertex2f(262.5+yy,330.0-xx);
                glEnd();
                if(xx>130){xx=130;yy=130;}
                else{
                xx+=0.1;
                yy+=0.1;}
                }
                else{xx=-40;
                yy=-40;}}
                else{
            glColor3f(1.0,0.647,0.0);//satellite
            glBegin(GL_POLYGON);//core
                glVertex2f(237.5,350.0);
                glVertex2f(252.5,350.0);
                glVertex2f(252.5,320.0);
                glVertex2f(237.5,320.0);
            glEnd();
            glColor3f(1.0,1.0,1.0);
            glBegin(GL_POLYGON);//side-panels
                glVertex2f(237.5,340.0);
                glVertex2f(230,340.0);
                glVertex2f(230,330.0);
                glVertex2f(237.5,330.0);

                glVertex2f(262.5,340.0);
                glVertex2f(227.5,340.0);
                glVertex2f(227.5,330.0);
                glVertex2f(262.5,330.0);
            glEnd();}
            }

            else{
            glColor3f(1.0,1.0,1.0);//bonnet
            glBegin(GL_POLYGON);//front
            glVertex2f(237.5,120.0+i);
            glVertex2f(262.5,120.0+i);
```



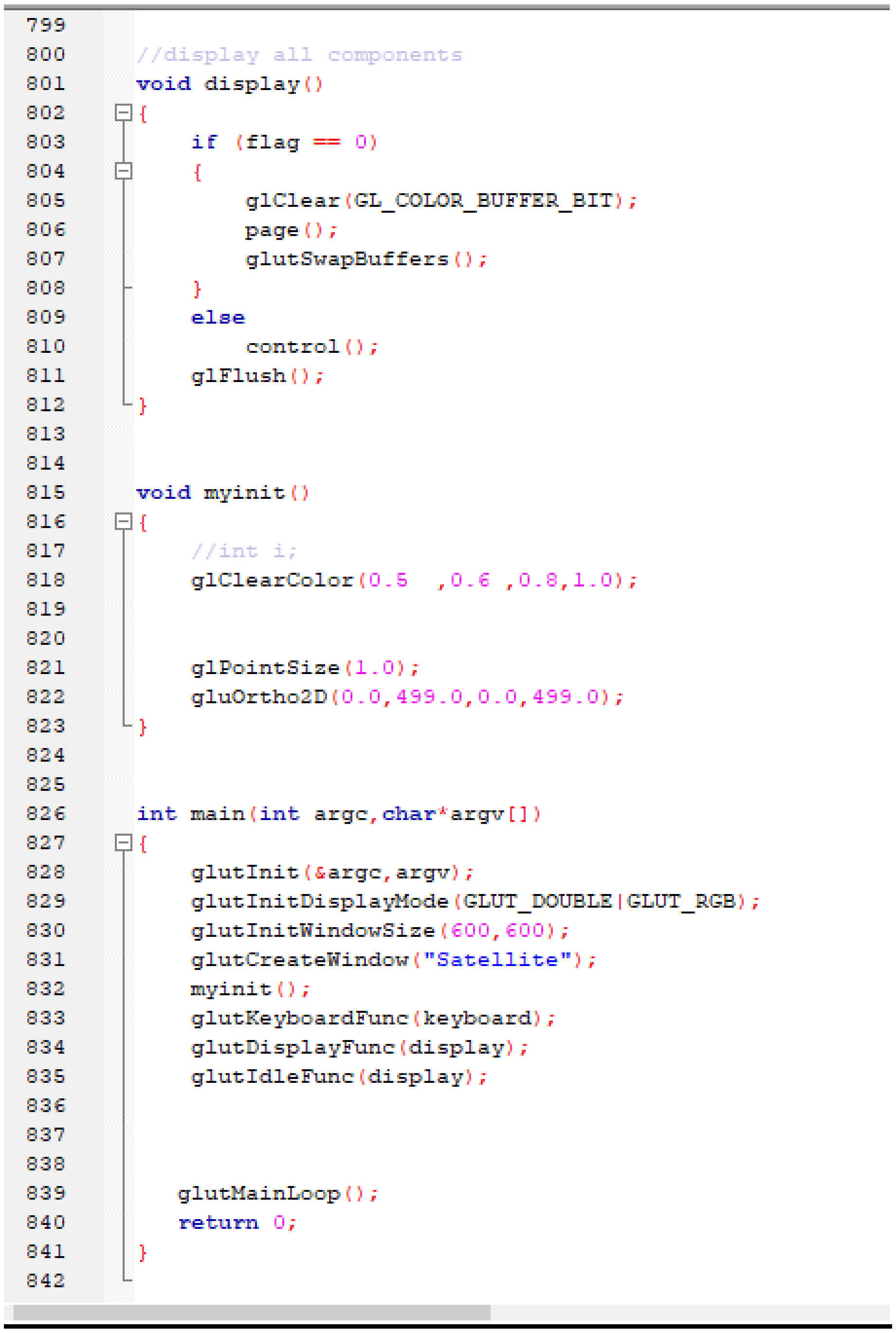
```
        glVertex2f(237.5,120.0+i);
        glVertex2f(262.5,120.0+i);
        glVertex2f(250,170.0+i);
        glEnd();
        }

        if(count<=120){
        glColor3f(1.0,0.0,0.0);
        glBegin(GL_POLYGON);//left_side_top
        glVertex2f(237.5,120.0+i);
        glVertex2f(217.5,95.0+i);
        glVertex2f(237.5,95.0+i);
        glEnd();
            glBegin(GL_POLYGON);//left_side_bottom
        glVertex2f(237.5,20.0+i);
        glVertex2f(217.5,20.0+i);
        glVertex2f(237.5,70.0+i);
        glEnd();
            glBegin(GL_POLYGON);//right_side_bottom
        glVertex2f(262.5,20.0+i);
        glVertex2f(282.5,20.0+i);
        glVertex2f(262.5,70.0+i);
        glEnd();
            glBegin(GL_POLYGON);//right_side_top
        glVertex2f(262.5,120.0+i);
        glVertex2f(262.5,95.0+i);
        glVertex2f(282.5,95.0+i);
        glEnd();
        }

        if(count<=110){
        glColor3f(0.556863 ,0.137255  ,0.419608);
            glBegin(GL_POLYGON);//bottom_1_exhaust
        glVertex2f(237.5,20.0+i);
        glVertex2f(244.5,20.0+i);
        glVertex2f(241,0.0+i);
        glEnd();
            glBegin(GL_POLYGON);//bottom_2_exhaust
        glVertex2f(246.5,20.0+i);
        glVertex2f(253.5,20.0+i);
        glVertex2f(249.5,0.0+i);
        glEnd();
            glBegin(GL_POLYGON);//bottom_3_exhaust
        glVertex2f(262.5,20.0+i);
```



```
            glBegin(GL_POLYGON);//bottom_3_exhaust
        glVertex2f(262.5,20.0+i);
        glVertex2f(255.5,20.0+i);
        glVertex2f(258.5,0.0+i);
        glEnd();
        }

    for(j=0;j<=1000000;j++)
        ;
    glutSwapBuffers();
    glutPostRedisplay();
    glFlush();
}
}

void mars(float radius)
{

    glColor3f(1,0,0);
   glBegin(GL_POLYGON);

   for (int i=0; i<=359; i++)
   {
      float degInRad = i*DEG2RAD;
      glVertex2f((300+f+cos(degInRad)*radius),(500-t+(sin(degInRad))*radius));
   }

   glEnd();
   t=t+0.1;
   if(t>200){
    t=210;

   }
}

//keys that trigger manual Lanch
void keyboard(unsigned char key, int x, int y)
{
    if (key == 'S' || key == 's'){
        for(int i=0;i<20000;i++)
            static_rocket();
        flag = 1;

    }
```



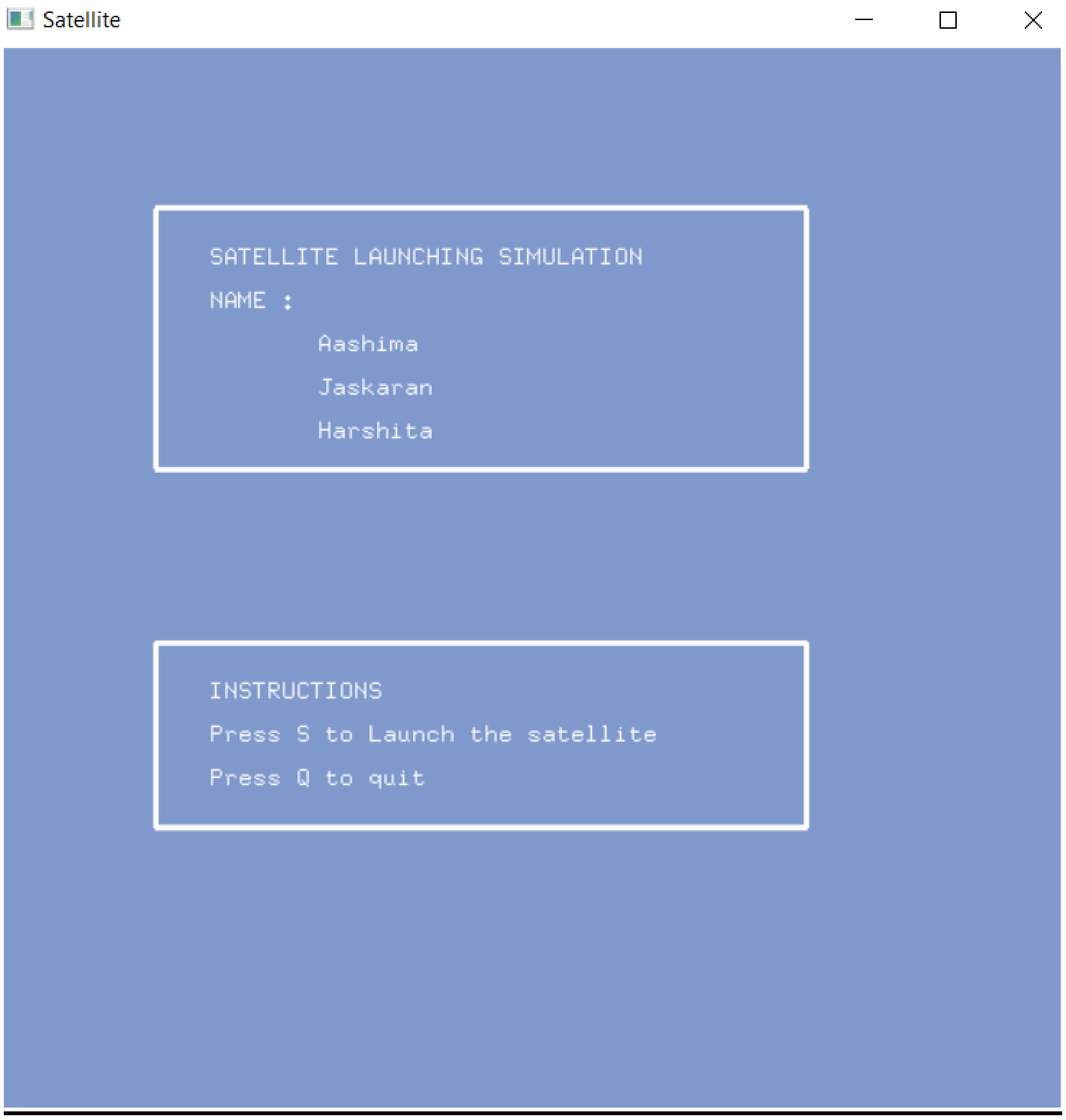
```
    }


    if (key == 'Q' || key == 'q')
        exit(0);

}

//design of homescreen
void page()
{
    glColor3f(1, 1, 1);
    glLineWidth(3);
    glBegin(GL_LINE_LOOP);
    glVertex2d(75, 425);
    glVertex2d(375, 425);
    glVertex2d(375, 305);
    glVertex2d(75, 305);
    glEnd();

    drawstring(100, 400, "SATELLITE LAUNCHING SIMULATION");
    drawstring(100, 380, "NAME : ");
    drawstring(150, 360, "Aashima");
    drawstring(150, 340, "Jaskaran");
    drawstring(150, 320, "Harshita");

    glBegin(GL_LINE_LOOP);
    glVertex2d(75, 140);
    glVertex2d(375, 140);
    glVertex2d(375, 225);
    glVertex2d(75, 225);
    glEnd();

    drawstring(100, 200, "INSTRUCTIONS");
    drawstring(100, 180, "Press S to Launch the satellite");
    drawstring(100, 160, "Press Q to quit");
    glFlush();
}

//display all components
void display()
{
    if (flag == 0)
```



```
//display all components
void display()
{
    if (flag == 0)
    {
        glClear(GL_COLOR_BUFFER_BIT);
        page();
        glutSwapBuffers();
    }
    else
        control();
    glFlush();
}


void myinit()
{
    //int i;
    glClearColor(0.5  ,0.6 ,0.8,1.0);


    glPointSize(1.0);
    gluOrtho2D(0.0,499.0,0.0,499.0);
}


int main(int argc,char*argv[])
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
    glutInitWindowSize(600,600);
    glutCreateWindow("Satellite");
    myinit();
    glutKeyboardFunc(keyboard);
    glutDisplayFunc(display);
    glutIdleFunc(display);



   glutMainLoop();
   return 0;
}

```

SCREENSHOTS:

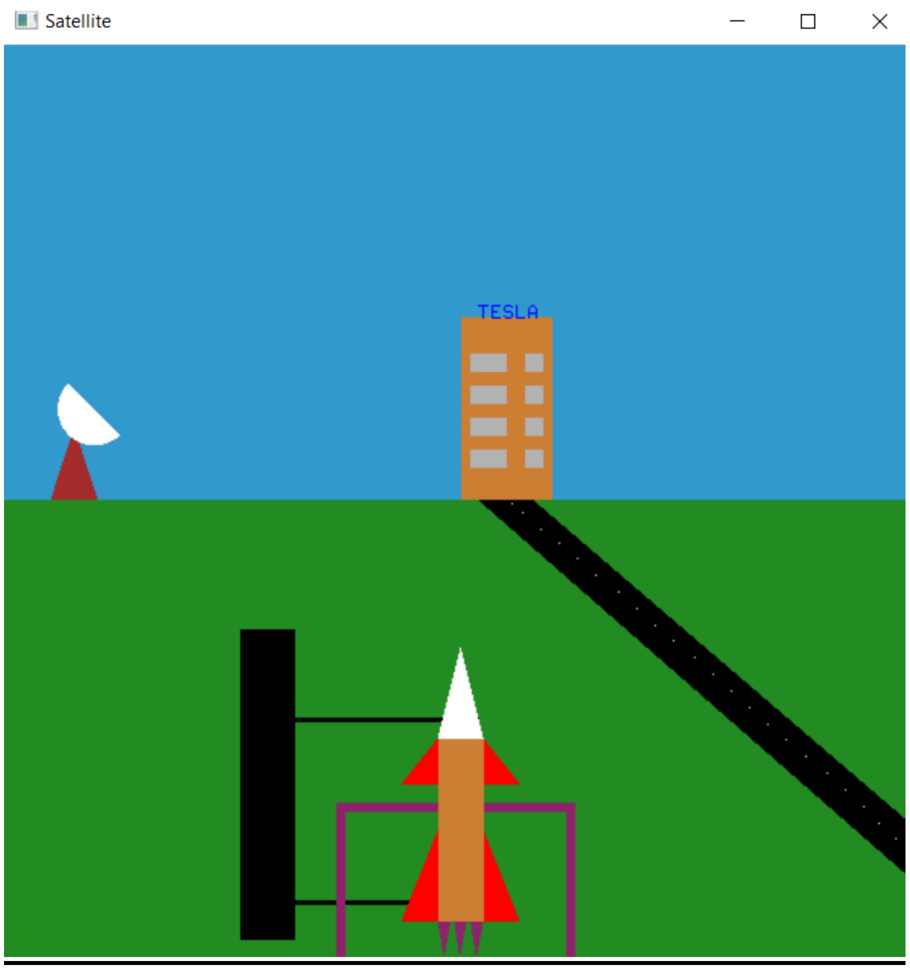
- Title Screen/Menu



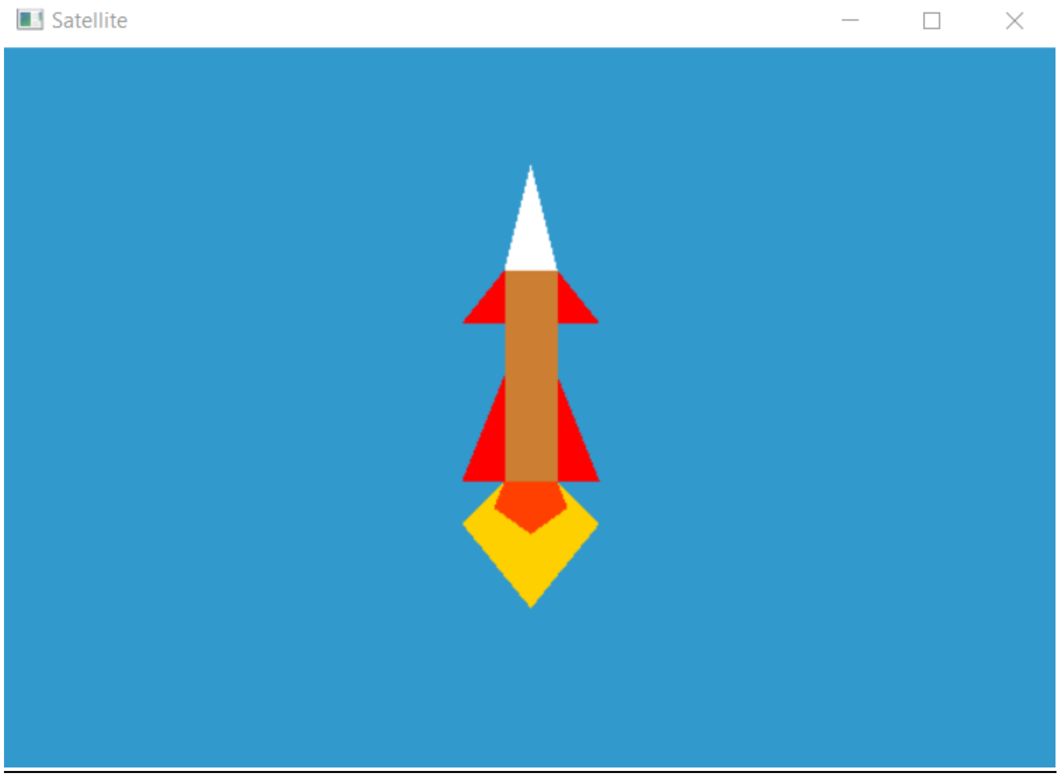
```
SATELLITE LAUNCHING SIMULATION
NAME :
          Aashima
          Jaskaran
          Harshita
```

```
INSTRUCTIONS
Press S to Launch the satellite
Press Q to quit
```

- <u>Initial Scene Before Launch:</u>



- <u>Atmosphere Flight Scene:</u>



- <u>Planet Scene with Tesla Cameo:</u>